TÉCNICAS DE ARITMÉTICA PARA LAS DERIVACIONES

- Rangos con una sola variable natural de la forma “$0 \leq i < n + 1$” se pueden trabajar de dos maneras.
    1. Por debajo, “$i = 0 \vee 1 \leq i < n + 1$”.
    2. Por arriba, “ $0 \leq i < n \vee i = n$”.

    Existe un caso particular, el cual es de la forma “$0 \leq i \leq n + 1$” se debe trabajar de la siguiente manera : “$0 \leq i \leq n \vee i = n + 1$”.
- Rangos con segmentos de lista de la forma “(x.xs) = as ++ bs ++ cs”, se trabajan mediante la propiedad de tercero excluido “(as = [ ] v as ≠ [ ])”. Cabe destacar que esta propiedad se aplica siempre al segmento inicial de los cuales conforman xs.
- Rangos con pares de elementos naturales de la forma “$0 \leq i < j < \#xs + 1$”. Estos se trabajan aritméticamente tal que “ $(i = 0 \vee i < 1) \wedge (i < j < \#xs + 1)$, luego hacemos distributividad y partición de rango, quedándonos de la forma “$(i = 0 \wedge i < j < \#xs + 1) \vee (1 \leq i < j < \#xs + 1)$”.
    - ❖ Un caso particular de este tipo de rangos es “$0 \leq i < j < n + 1$”, que se trabaja aritméticamente por arriba, tal que “$0 \leq i < j \wedge (j = n \vee j < n)$”, luego se aplica distributividad y partición de rango de la misma forma que el anterior.

Nota : durante las derivaciones, siempre tratar de trabajar primero el término sobre el cual se aplica la hipótesis inductiva, por que, en caso de necesitar una generalización, la derivación anterior no sirve como prueba.
Además, en la programación imperativa, conviene siempre comenzar con una refinación del ciclo para tener una idea a nivel estructural, y luego derivar el cuerpo del mismo, porque, en caso de necesitar un refuerzo de invariante, tenemos que volver a probar todo.

Derivaciones en el paradigma Funcional

- h.xs = $< \exists\ k : 0 \leq k < \#xs : sum(xs \uparrow k) > k >$

Derivemos h mediante inducción en xs con el caso base xs = [ ] y el caso inductivo xs = (x.xs).

$$h.[\,] = < \exists\ k : 0 \leq k < \#[\,] : sum([\,] \uparrow k) > k >$$
{Definición de cardinal de lista}
$$h.[\,] = < \exists\ k : 0 \leq k < 0 : sum(xs \uparrow k) > k >$$
{Aritmética}
$$h.[\,] = < \exists\ k : False : sum(xs \uparrow k) > k >$$
{Rango vacío}
$$h.[\,] = False$$

$$h.(x.xs) = < \exists\ k : 0 \leq k < \#(x.xs) : sum((x.xs) \uparrow k) > k >$$
{Definición de cardinal}
$$h.(x.xs) = < \exists\ k : 0 \leq k < 1 + \#xs : sum((x.xs) \uparrow k) > k >$$
{Aritmética en el rango}
$$h.(x.xs) = < \exists\ k :ki = 0 \vee 1 \leq k < 1 + \#xs : sum((x.xs) \uparrow k) > k >$$
{Partición de rango}
$$h.(x.xs) = < \exists\ k : 1 \leq k < 1 + \#xs : sum((x.xs) \uparrow k) > k > \vee$$
$$< \exists\ k : k= 0 : sum((x.xs) \uparrow k) > k >$$
{Cambio de variable k ← k + 1, rango unitario}
$$h.(x.xs) = < \exists\ k : 1 \leq k + 1 < 1 + \#xs : sum((x.xs) \uparrow k + 1) > k + 1 > \vee$$
$$(sum((x.xs) \uparrow 0) > 0)$$
{Definición de ↑, aritmética en el rango}
$$h.(x.xs) = < \exists\ k : 0 \leq k < \#xs : sum(x . (xs \uparrow k)) > k + 1 > \vee$$
$$(sum.[\,] > 0)$$
{Definición de sum.[ ] y de sum(x . (xs ↑ k))}
$$h.(x.xs) = < \exists\ k : 0 \leq k < \#xs : x + sum.(xs \uparrow k) > k + 1 > \vee$$
$$(0 > 0)$$
{Aritmética y elemento neutro de la disyunción}
$$h.(x.xs) = < \exists\ k : 0 \leq k < \#xs : x + sum.(xs \uparrow k) > k + 1 >$$
{No puedo aplicar hipótesis inductiva, debo generalizar}

genh.r.p.xs = $< \exists\ k : 0 \leq k < \#xs : r + sum.(xs \uparrow k) > k + p >$
Ahora probemos que h es un caso particular de la función generalizada.

$$genh.r.p.xs = < \exists\ k : 0 \leq k < \#xs : r + sum.(xs \uparrow k) > k + p >$$
{Propuesta de r ← 0 y p ← 0}
$$genh.0.0.xs = < \exists\ k : 0 \leq k < \#xs : 0 + sum.(xs \uparrow k) > k + 0 >$$

{Elemento neutro de la suma}

$$genh.0.0.xs = \langle \exists\ k : 0 \leq k < \#xs : sum.(xs \uparrow k) > k \rangle$$

{Definición de h.xs}

$$genh.0.0.xs = h.xs$$

Derivemos la función generalizada :

$$genh.r.p.[\ ] = \langle \exists\ k : 0 \leq k < \#[\ ] : r + sum.([\ ] \uparrow k) > k + p \rangle$$

{Definición de cardinal}

$$genh.r.p.[\ ] = \langle \exists\ k : 0 \leq k < 0 : r + sum.([\ ] \uparrow k) > k + p \rangle$$

{Aritmética en el rango}

$$genh.r.p.[\ ] = \langle \exists\ k : False : r + sum.([\ ] \uparrow k) > k + p \rangle$$

{Rango vacío}

$$genh.r.p.[\ ] = False$$

$$genh.r.p.(x.xs) = \langle \exists\ k : 0 \leq k < \#(x.xs) : r + sum.((x.xs) \uparrow k) > k + p \rangle$$

{Definición de cardinal}

$$genh.r.p.(x.xs) = \langle \exists\ k : 0 \leq k < 1 + \#xs : r + sum.((x.xs) \uparrow k) > k + p \rangle$$

{Aritmética en el rango}

$$genh.r.p.(x.xs) = \langle \exists\ k : k = 0 \vee 1 \leq k < 1 + \#xs : r + sum.((x.xs) \uparrow k) > k + p \rangle$$

{Partición de rango}

$$genh.r.p.(x.xs) = \langle \exists\ k : 1 \leq k < 1 + \#xs : r + sum.((x.xs) \uparrow k) > k + p \rangle \vee$$
$$\langle \exists\ k : k = 0 : r + sum.((x.xs) \uparrow k) > k + p \rangle$$

{Cambio de variable $k \leftarrow k + 1$, rango unitario}

$$genh.r.p.(x.xs) = \langle \exists\ k : 1 \leq k + 1 < 1 + \#xs : r + sum.((x.xs) \uparrow k + 1) > k + 1 + p \rangle \vee$$
$$r + sum.((x.xs) \uparrow 0) > 0 + p$$

{Definición de take, aritmética en el rango y neutro de la suma}

$$genh.r.p.(x.xs) = \langle \exists\ k : 0 \leq k < \#xs : r + sum.(x.(xs \uparrow k)) > k + 1 + p \rangle \vee$$
$$r + sum.[\ ] > p$$

{Definición de sum y aritmética}

$$genh.r.p.(x.xs) = \langle \exists\ k : 0 \leq k < \#xs : (r + x) + sum.(xs \uparrow k) > k + (1 + p) \rangle \vee (r > p)$$

{Hipótesis inductiva}

$$genh.r.p.(x.xs) = genh.(r + x).(1 + p).(x.xs) \vee (r > p)$$

Programa final, anotado :

h.[ ] = False

h.(x.xs) = genh.0.0.xs

genh.r.p.[ ] = False

genh.r.p.(x.xs) = genh.(r + x).(1 + p).(x.xs) v (r > p)

- $f.xs = \langle \Sigma\, i, j : 0 \le i \le j < \#xs : xs.i * xs.j \rangle$

Derivemos mediante inducción en xs. El caso base es xs = [ ] y el caso inductivo es xs = (x.xs).

$$f.[\,] = \langle \Sigma\, i, j : 0 \le i \le j < \#[\,] : [\,].i * [\,].j \rangle$$
{Definición de cardinal de lista vacía}
$$f.[\,] = \langle \Sigma\, i, j : 0 \le i \le j < 0 : [\,].i * [\,].j \rangle$$
{Aritmética}
$$f.[\,] = \langle \Sigma\, i, j : False : [\,].i * [\,].j \rangle$$
{Rango vacío}
$$f.[\,] = 0$$

$$f.(x.xs) = \langle \Sigma\, i, j : 0 \le i \le j < \#(x.xs) : (x.xs).i * (x.xs).j \rangle$$
{Definición de cardinal de lista}
$$f.(x.xs) = \langle \Sigma\, i, j : 0 \le i \le j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle$$
{Aritmética en el rango}
$$f.(x.xs) = \langle \Sigma\, i, j : (i = 0 \lor 1 <= i) \land i \le j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle$$
{Distributividad y partición de rango}
$$f.(x.xs) = \langle \Sigma\, i, j : i = 0 \land i \le j < 1 + \#xs : xs.i * xs.j \rangle + \langle \Sigma\, i, j : 1 <= i \le j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle$$
{Eliminación de variable con i = 0, cambio de variables, i ← i + 1 y j ← j + 1}
$$f.(x.xs) = \langle \Sigma\, j : 0 \le j < 1 + \#xs : (x.xs).0 * (x.xs).j \rangle + \langle \Sigma\, i, j : 1 <= i + 1 \le j + 1 < 1 + \#xs : (x.xs).i + 1 * (x.xs).j + 1 \rangle$$
{Aritmética en el rango y definición de indexación}
$$f.(x.xs) = \langle \Sigma\, j : 0 \le j < 1 + \#xs : x * (x.xs).j \rangle + \langle \Sigma\, i, j : 0 <= i \le j < \#xs : xs.i * xs.j \rangle$$
{Aritmética en el rango e hipótesis inductiva}
$$f.(x.xs) = \langle \Sigma\, j : j = 0 \lor 1 \le j < 1 + \#xs : x * (x.xs).j \rangle + f.xs$$
{Partición de Rango}
$$f.(x.xs) = \langle \Sigma\, j : 1 \le j < 1 + \#xs : x * (x.xs).j \rangle + \langle \Sigma\, j : j = 0 : x * (x.xs).j \rangle + f.xs$$
{Rango unitario}
$$f.(x.xs) = \langle \Sigma\, j : 1 \le j < 1 + \#xs : x * (x.xs).j \rangle + (x * (x.xs).0) + f.xs$$
{Definición de indexación}
$$f.(x.xs) = \langle \Sigma\, j : 1 \le j < 1 + \#xs : x * (x.xs).j \rangle + (x * x) + f.xs$$
{Cambio de variable j ← j + 1}
$$f.(x.xs) = \langle \Sigma\, j : 1 \le j + 1 < 1 + \#xs : x * (x.xs).j + 1 \rangle + (x * x) + f.xs$$
{Aritmética en el rango y definición de indexación}
$$f.(x.xs) = \langle \Sigma\, j : 0 \le j < \#xs : x * xs.j \rangle + (x * x) + f.xs$$
{Modularización}
$$f.(x.xs) = mod1.k.xs + (x * x) + f.xs$$

Derivemos mod1.k.xs mediante inducción en xs con el caso base xs = [ ] y el caso inductivo xs = (x.xs).

$$mod1.k.[\,] = \langle \Sigma\, j : 0 \leq j < \#[\,] : k * [\,].j \rangle$$
{Definición de cardinal de lista vacía}
$$mod1.k.[\,] = \langle \Sigma\, j : 0 \leq j < 0 : k * [\,].j \rangle$$
{Aritmética en el el rango y rango vacío}
$$mod1.k.[\,] = 0$$

$$mod1.k.(x.xs) = \langle \Sigma\, j : 0 \leq j < \#(x.xs) : k * (x.xs).j \rangle$$
{Definición de cardinal de lista}
$$mod1.k.(x.xs) = \langle \Sigma\, j : 0 \leq j < 1 + \#xs : k * (x.xs).j \rangle$$
{Aritmética en el rango}
$$mod1.k.(x.xs) = \langle \Sigma\, j : j = 0 \lor 1 \leq j < 1 + \#xs : k * (x.xs).j \rangle$$
{Separación de un término}
$$k* (x.xs).0 + \langle \Sigma\, j : 1 \leq j < 1 + \#xs : k * (x.xs).j \rangle$$
{Definición de indexación, cambio de variable $j \leftarrow j + 1$}
$$(k * x) + \langle \Sigma\, j : 1 \leq j + 1 < 1 + \#xs : k * (x.xs).j + 1 \rangle$$
{Aritmética en el rango, definición de indexación}
$$(k * x) + \langle \Sigma\, j : 0 \leq j < \#xs : k * xs.j \rangle$$
{Hipótesis inductiva}
$$(k * x) + mod1.k.xs$$

Programa final, anotado:
f.[ ] = 0
f.(x.xs) = mod1.k.xs + (x ∗ x) + f.xs
mod1.k.[ ] = 0
mod1.k.xs = (k ∗ x) + mod1.k.xs

- $f.xs = \langle \exists\, i,j : 0 <= i < j < \#xs : xs.i = xs.j \rangle$

Caso base con xs = [ ]
$$f.[\,] = \langle \exists\, i,j : 0 <= i < j < \#[\,] : [\,].i = [\,].j \rangle$$
{Definición de cardinal}
$$f.[\,] = \langle \exists\, i,j : 0 <= i < j < 0 : [\,].i = [\,].j \rangle$$
{Aritmética en el rango}
$$f.[\,] = \langle \exists\, i,j : False : [\,].i = [\,].j \rangle$$
{Rango vacío}
$$False$$

Caso inductivo con xs = (x.xs)
$$f.(x.xs) = \langle \exists\, i,j : 0 <= i < j < \#(x.xs) : (x.xs).i = (x.xs).j \rangle$$
{Definición de cardinal de lista}
$$f.(x.xs) = \langle \exists\, i,j : 0 <= i < j < 1 + \#xs : (x.xs).i = (x.xs).j \rangle$$
{Aritmética en el rango}
$$f.(x.xs) = \langle \exists\, i,j : (0 = i \lor 1 <= i) \wedge i < j < 1 + \#xs : (x.xs).i = (x.xs).j \rangle$$
{Distributividad en el rango}

$f.(x.xs) = < \exists\, i,j : (0 = i \wedge i < j < 1 + \#xs) \vee (1 <= i \wedge i < j < 1 + \#xs) : (x.xs).i = (x.xs).j >$

{Partición de rango}

$f.(x.xs) = < \exists\, i,j : (0 = i \wedge i < j < 1 + \#xs) : (x.xs).i = (x.xs).j > \vee$
$< \exists\, i,j : (1 <= i \wedge i < j < 1 + \#xs) : (x.xs).i = (x.xs).j >$

{Eliminación de variable en el primer término con i = 0}

$f.(x.xs) = < \exists\, j : 0 < j < 1 + \#xs : (x.xs).0 = (x.xs).j > \vee$
$< \exists\, i,j : (1 <= i \wedge i < j < 1 + \#xs) : (x.xs).i = (x.xs).j >$

{Aritmética en el rango de la segunda cuantificación, cambio de variable i ← i + 1, j ← j + 1}

$f.(x.xs) = < \exists\, j : 0 < j < 1 + \#xs : (x.xs).0 = (x.xs).j > \vee$
$< \exists\, i,j : (1 <= i + 1 < j + 1 < 1 + \#xs) : (x.xs).i + 1 = (x.xs).j + 1 >$

{Definición de indexación, aritmética en el rango}

$f.(x.xs) = < \exists\, j : 1 <= j < 1 + \#xs : x = (x.xs).j > \vee$
$< \exists\, i,j : 0 <= i < j < \#xs : xs.i = xs.j >$

{Cambio de variable j ← j + 1, hipótesis inductiva}

$f.(x.xs) = < \exists\, j : 1 <= j + 1 < 1 + \#xs : x = (x.xs).j + 1 > \vee f.xs$

{Aritmética en el rango y definición de indexación}

$f.(x.xs) = < \exists\, j : 0 <= j < \#xs : x = xs.j > \vee f.xs$

{Modularizacion existe.k.xs}

$f.(x.xs) = existe.k.xs \vee f.xs$

Ahora derivemos existe.x.xs, nuevamente mediante inducción.

Caso base con xs = [ ]

existe.k.[ ]

{Especificación}

$< \exists\, j : 0 <= j < \#[\,] : k = xs.j >$

{Cardinal de lista vacía}

$< \exists\, j : 0 <= j < 0 : k = xs.j >$

{Aritmética}

$< \exists\, j : False : k = xs.j >$

{Rango vacío}

False

Caso inductivo con xs = (x.xs)

existe.x.(x.xs)

{Especificación}

$< \exists\, j : 0 <= j < \#(x.xs) : k = (x.xs).j >$

{Cardinal de lista}

$< \exists\, j : 0 <= j < 1 + \#xs : k = (x.xs).j >$

{Aritmética}

$< \exists\, j : j = 0 \vee 1 <= j < 1 + \#xs : k = (x.xs).j >$

{Partición de Rango}

$< \exists\, j : j = 0 : k = (x.xs).j > \vee < \exists\, j : 1 <= j < 1 + \#xs : k = (x.xs)j >$

{Rango unitario en el primer término, cambio de variable en el segundo}

k = (x.xs).0 v < ∃ j : 1 <= j + 1 < 1 + #xs : k = (x.xs)j + 1 >
{Definición de indexación y aritmética en el rango}
k = x v < ∃ j : 0 <= j < #xs : k = xs.j >
{Hipótesis Inductiva}
k = x v existe.k.xs

Programa final : f.[ ] = False
f. (x.xs) = existe.x.xs v f.xs
existe.k.[ ] = False
existe.k.(x.xs) = (k = x) v existe.k.xs

- f.xs = < N as,bs : xs = as ++ bs : sum.as / (#as + 1) = 8 >

Derivemos este programa mediante inducción en xs.

Caso Base con xs = [ ]
f. [ ]
{Especificación}
< N as,bs : [ ] = as ++ bs : sum.as / (#as + 1) = 8 >
{Propiedad de listas}
< N as,bs : [ ] = as ^ [ ] = bs : sum.as / (#as + 1) = 8 >
{Definición de conteo}
< Σ as,bs : [ ] = as ^ [ ] = bs ^ sum.as / (#as + 1) = 8 : 1 >
{Anidado}
< Σ as : [ ] = as : < Σ bs : [ ] = bs ^ sum.as / (#as + 1) = 8 : 1 > >
{Rango Unitario}
< Σ bs : [ ] = bs ^ sum.[ ] / (#[ ] + 1) = 8 : 1 >
{Definición de sum.xs y cardinal de lista vacía}
< Σ bs : [ ] = bs ^ (0 / (0 + 1) = 8) : 1 >
{Aritmética}
< Σ bs : [ ] = bs ^ False: 1 >
{Absorbente de la conjunción}
< Σ bs : False : 1 >
{Rango vacío}
0
Caso Inductivo con xs = (x.xs)

f.(x.xs)
{Especificación}
< N as,bs : (x.xs) = as ++ bs : sum.as / (#as + 1) = 8 >
{Tercero excluido}
< N as,bs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ bs : sum.as / (#as + 1) = 8 >
{Definición de conteo}
< Σ as,bs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ bs ^ sum.as / (#as + 1) = 8 : 1 >

{Distributividad de v con ^}

< Σ as,bs : (as = [ ] ^ (x.xs) = as ++ bs ^ sum.as / (#as + 1)) = 8 v (as ≠ [ ] ^ (x.xs) = as ++ bs ^ sum.as / (#as + 1) = 8 : 1 >

{Partición de rango}

< Σ as,bs : as = [ ] ^ (x.xs) = as ++ bs ^ sum.as / (#as + 1) = 8 : 1 > +
< Σ as,bs: as ≠ [ ] ^ (x.xs) = as ++ bs ^ sum.as / (#as + 1) = 8 : 1 >

{Anidado en el primer término y cambio de variable as = (a.as) válido por as distinto de vacia}

< Σ as: as = [ ] **:** < Σbs :(x.xs) = as ++ bs ^ sum.as / (#as + 1) = 8 : 1 >> +
< Σ a,as,bs: (a.as) ≠ [ ] ^ (x.xs) = (a.as) ++ bs ^ sum.(a.as) / (#(a.as) + 1) = 8 : 1 >

{Rango unitario en el primer término, aritmética en el segundo y definición de sum.xs, cardinal de lista}

< Σbs :(x.xs) = [ ] ++ bs ^ 0 / (1) = 8 : 1 > +
< Σ a, as,bs: True ^ (x.xs) = (a.as) ++ bs ^ a + sum.as / ((1 + #as) + 1) = 8 : 1 >

{Propiedad de listas,aritmética}

< Σbs :(x.xs) = bs ^ 0 = 8 : 1 > +
< Σ a, as,bs: True ^ (x.xs) = a.(as ++ bs) ^ a + sum.as / ((1 + #as) + 1) = 8 : 1 >

{Aritmética}

< Σbs :(x.xs) = bs ^ False : 1 > +
< Σ a,as,bs : (x.xs) = a.(as ++ bs) ^ a + sum.as / ((1 + #as) + 1) = 8 : 1 >

{Absorbente de la conjunción}

< Σbs : False : 1 > +
< Σ a,as,bs : (x.xs) = a.(as ++ bs) ^ a + sum.as / ((1 + #as) + 1) = 8 : 1 >

{Rango vacío, propiedad de listas}

0 +
< Σ a,as,bs : x = a ^ xs = as ++ bs ^ a + sum.as / ((1 + #as) + 1) = 8 : 1 >

{Aritmética, anidado}

< Σ a : a = x : < Σ as,bs :xs = as ++ bs ^ a + sum.as / ((1 + #as) + 1) = 8 : 1 >>

{Rango unitario}

< Σ as,bs :xs = as ++ bs ^ x + sum.as / ((1 + #as) + 1) = 8 : 1 >

{Definición de conteo}

< N as,bs : xs = as ++ bs : x + sum.as / ((1 + #as) + 1) = 8 >

{Mi hipótesis inductiva es muy rígida, por lo que propongo una generalización por abstracción}

genf.k.p.xs = < N as,bs : xs = as ++ bs : k + sum.as / ((1 + #as) + p) = 8 >

Por lo tanto mi programa f.xs está dado por genf.0.0.xs, veámoslo :

genf.0.0.xs

{Especificación}

< N as,bs : xs = as ++ bs : 0 + sum.as / ((1 + #as) + 0) = 8 >

{Aritmética}

< N as,bs : xs = as ++ bs : sum.as / (1 + #as) = 8 >

{Especificación de f.xs}

f.xs

Bien, ahora toca derivar genf.k.p.xs, mediante inducción en xs.

Caso Base con xs = [ ]

genf.k.p.[ ] = < N as,bs : [ ] = as ++ bs : k + sum.as / ((1 + #as) + p) = 8 >
{Propiedad de listas}
genf.k.p.[ ] = < N as,bs : [ ] = as ^ [ ] = bs : k + sum.as / ((1 + #as) + p) = 8 >
{Eliminación de variable con as}
genf.k.p.[ ] = < N bs : [ ] = bs : k + sum.[ ] / ((1 + #[ ]) + p) = 8 >
{Definición de sum y de cardinal de listas}
genf.k.p.[ ] = < N bs : [ ] = bs : k + 0 / ((1 + 0) + p) = 8 >
{Neutro de la suma}
genf.k.p.[ ] = < N bs : [ ] = bs : k / (1 + p) = 8 >
{Rango unitario}
genf.k.p.[ ] = (k / (1 + p) = 8 → 1
k / (1 + p) ≠ 8 → 0
)

Caso inductivo con xs = (x.xs)

genf.k.p.(x.xs) = < N as,bs : (x.xs) = as ++ bs : k + sum.as / ((1 + #as) + p) = 8 >
{Definición de conteo}
genf.k.p.(x.xs) = < Σ as,bs : (x.xs) = as ++ bs ^ k + sum.as / ((1 + #as) + p) = 8 : 1 >
{Tercero excluido}
genf.k.p.(x.xs) = < Σ as,bs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ bs ^ k + sum.as / ((1 + #as) + p) = 8 : 1 >
{Distributividad y partición de rango}
genf.k.p.(x.xs) = < Σ as,bs : as = [ ] ^ (x.xs) = as ++ bs ^ k + sum.as / ((1 + #as) + p) = 8 : 1 > + < Σ as,bs : as ≠ [ ] ^ (x.xs) = as ++ bs ^ k + sum.as / ((1 + #as) + p) = 8 : 1 >
{Eliminación de variable con as en el primer término}
genf.k.p.(x.xs) = < Σ bs : (x.xs) = [ ] ++ bs ^ k + sum.[ ] / ((1 + #[ ]) + p) = 8 : 1 > + < Σ as,bs : as ≠ [ ] ^ (x.xs) = as ++ bs ^ k + sum.as / ((1 + #as) + p) = 8 : 1 >
{Propiedades en el primer término, reemplazo de as por (a.as) en el segundo}
genf.k.p.(x.xs) = < Σ bs : (x.xs) = bs ^ k + 0 / ((1 + 0) + p) = 8 : 1 > +
< Σ a,as,bs : (x.xs) = (a.as) ++ bs ^ k + sum.(a.as) / ((1 + #(a.as)) + p) = 8 : 1 >
{Propiedades en ambos términos}
genf.k.p.(x.xs) = < Σ bs : (x.xs) = bs ^ k / (1 + p) = 8 : 1 > +
< Σ a,as,bs : (x.xs) = (a.as) ++ bs ^ k + (a + sum.as) / ((1 + (1 + #as) + p) = 8 : 1 >
{Definición de conteo en el primer término y propiedad de listas en el segundo}
genf.k.p.(x.xs) = < N bs : (x.xs) = bs : k / (1 + p) = 8 > +
< Σ a,as,bs : (x = a) ^ (xs = as ++ bs) ^ k + (a + sum.as) / (1 + (1 + #as) + p) = 8 : 1 >
{Eliminación de variable con a}
genf.k.p.(x.xs) = < N bs : (x.xs) = bs : k / (1 + p) = 8 > +
< Σ as,bs :xs = as ++ bs ^ k + (x + sum.as) / (1 + (1 + #as) + p) = 8 : 1 >

{Definición de conteo en el segundo término}
genf.k.p.(x.xs) = < N bs : (x.xs) = bs : k / (1 + p) = 8 > +
< N as,bs : xs = as ++ bs : k + (x + sum.as) / (1 + (1 + #as) + p) = 8 >
{Aritmética en el segundo término}
genf.k.p.(x.xs) = < N bs : (x.xs) = bs : k / (1 + p) = 8 > +
< N as,bs : xs = as ++ bs : (k + x) + sum.as / (1 + #as) + (1+ p)) = 8 >
{Hipótesis inductiva}
genf.k.p.(x.xs) = < N bs : (x.xs) = bs : k / (1 + p) = 8 > + genf.(k + x).(1 + p).xs
{Rango unitario}
(k / (1 + p) = 8 → 1 + genf.(k + x).(1 + p).xs
k / (1 + p) ≠ 8 → genf.(k + x).(1 + p).xs
)

Programa Final :
f. [ ] = 0
f. (x.xs) = genf.0.0.xs
genf.k.p.[ ] = (k / (1 + p) = 8 ---> 1
k / (1 + p) ≠ 8 ---> 0
)
genf.k.p.xs = (k / 1 + p = 8 ---> 1 + genf.(k + x).(p + 1)
k / 1 + p ≠ 8 ---> genf.(k + x).(p + 1)
)

- f.xs = < Σ i,j : 0 <= i <= j < #xs : xs.i * xs.j >

Caso Base con xs = [ ]
f. [ ]
{Especificación}
< Σ i,j : 0 <= i <= j <= #[ ] : [ ].i * [ ].j >
{Cardinal de lista vacía}
< Σ i,j : 0 <= i <= j <= 0 : [ ].i * [ ].j >
{Aritmética}
< Σ i,j : False : [ ].i * [ ].j >
{Rango vacío}
0
Caso Inductivo con xs = (x.xs)
f.(x.xs)
{Especificación}
< Σ i,j : 0 <= i <= j < #(x.xs) : (x.xs).i * (x.xs).j >
{Cardinal de lista}
< Σ i,j : 0 <= i <= j < 1 + #xs : (x.xs).i * (x.xs).j >
{Aritmética en el rango}
< Σ i,j : 0 <= i ^ i <= j < 1 + #xs : (x.xs).i * (x.xs).j >
{Aritmética en el rango}

$\langle \Sigma\, i,j : (0 = i \vee 0 < i) \wedge i <= j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle$

{Distributividad de v con ^}

$\langle \Sigma\, i,j : (0 = i \wedge i <= j < 1 + \#xs) \vee (0 < i \wedge i <= j < 1 + \#xs) : (x.xs).i * (x.xs).j \rangle$

{Partición de Rango}

$\langle \Sigma\, i,j : (0 = i \wedge i <= j < 1 + \#xs) : (x.xs).i * (x.xs).j \rangle + \langle \Sigma\, i,j : (0 < i \wedge i <= j < 1 + \#xs) : (x.xs).i * (x.xs).j \rangle$

{Anidado en el primer término, aritmética en el segundo término}

$\langle \Sigma\, i : 0 = i : \langle \Sigma\, j: i <= j < 1 + \#xs: (x.xs).i * (x.xs).j \rangle\rangle + \langle \Sigma\, i,j : 1 <= i <= j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle$

{Rango unitario en el primer término, cambio de variable en el segundo : $j \leftarrow j + 1$, $i \leftarrow i + 1$}

$\langle \Sigma\, j: 0 <= j < 1 + \#xs: (x.xs).0 * (x.xs).j \rangle + \langle \Sigma\, i,j : 1 <= i + 1 <= j + 1 < 1 + \#xs : (x.xs).i + 1 * (x.xs).j + 1 \rangle$

{Definición de indexación, aritmética en el rango del segundo término y definición de indexación}

$\langle \Sigma\, j: 0 <= j < 1 + \#xs: x * (x.xs).j \rangle + \langle \Sigma\, i,j : 0 <= i <= j < \#xs : xs.i * xs.j \rangle$

{Hipótesis Inductiva}

$\langle \Sigma\, j: 0 <= j < 1 + \#xs: x * (x.xs).j \rangle + f.xs$

{El primer término no es programa, por lo que modularizar,pero trabajo un poco mas el termino}

$\langle \Sigma\, j: 0 = j \vee 1 <= j < 1 + \#xs: x * (x.xs).j \rangle + f.xs$

{Partición de Rango}

$\langle \Sigma\, j: 0 = j: x * (x.xs).j \rangle + \langle \Sigma\, j: 1 <= j < 1 + \#xs: x * (x.xs).j \rangle + f.xs$

{Rango unitario}

$(x * x) + \langle \Sigma\, j: 1 <= j < 1 + \#xs: x * (x.xs).j \rangle + f.xs$

{Cambio de variable $j \leftarrow j + 1$ y aritmética}

$(x * x) + \langle \Sigma\, j: 0 <= j < \#xs: x * (x.xs).j + 1 \rangle + f.xs$

{Definición de indexación}

$(x * x) + \langle \Sigma\, j: 0 <= j < \#xs: x * xs.j \rangle + f.xs$

{Modularizo}

$(x * x) + sumCuad.xs + f.xs$

Ahora derivemos sumCuad.xs nuevamente por inducción.

Caso Base con xs = [ ]

sumCuad.[ ]

{Especificación}

$\langle \Sigma\, j: 0 <= j < \#[\ ]: x * xs.j \rangle$

{Cardinal de lista vacía y rango vacío}

0

Caso Base con xs = (x.xs)

sumCuad.(x.xs)

{Especificación}

$\langle \Sigma\, j: 0 <= j < \#(x.xs): x * (x.xs).j \rangle$

{Cardinal de lista}

<Σ j: 0 <= j < 1 + #xs: x * (x.xs).j >
{Aritmética en el rango}
<Σ j: (0 = j v 1 <= j < 1 + #xs) : x * (x.xs).j >
{Partición de Rango}
<Σ j: 0 = j : x * (x.xs).j > + <Σ j: 1 <= j < 1 + #xs : x * (x.xs).j >
{Rango unitario, cambio de variable j ← j + 1 y aritmética en el rango}
(x * x) + <Σ j: 0 <= j < #xs : x * (x.xs).j + 1 >
{Definición de indexación}
(x * x) + <Σ j: 0 <= j < #xs : x * xs.j >
{Hipótesis Inductiva}
(x * x) + sumCuad.xs

Finalmente, el programa derivado:
f.[ ] = 0
f.(x.xs) = (x * x) + sumCuad.xs + f.xs
sumCuad.[ ] = 0
sumCuad.(x.xs) = (x * x) + sumCuad.xs

- sum pot.x.n = < Σ i : 0 ≤ i < n : x^i >

Hacemos inducción en “n”, caso base con n = 0 y caso inductivo con n = n + 1.

sum pot.x.0 = < Σ i : 0 ≤ i < 0 : x^i >
{Aritmética y lógica en el rango}
sum pot.x.0 = < Σ i : False : x^i >
{Rango vacío de sumatoria}
sum pot.x.0 = 0

sum pot.x.(n+1) = < Σ i : 0 ≤ i < n + 1 : x^i >
{Aritmética en el rango}
sum pot.x.(n+1) = < Σ i : 0 ≤ i < n v (i = n) : x^i >
{Partición de rango}
sum pot.x.(n+1) = < Σ i : 0 ≤ i < n : x^i > + < Σ i : i = n : x^i >
{Hipótesis inductiva y rango unitario}
sum pot.x.(n+1) = sum pot.x.n + x^n
{Modularizo x^n}
sum pot.x.(n+1) = sum pot.x.n + exp.x.n

Ahora debemos derivar exp.x.n = x ^ n. Hacemos inducción en n.

exp.x.0 = x ^ 0
{Propiedad de potencias}
1

$$exp.x.(n+1) = x ^ (n+1)$$
{Propiedad de potencias}
$$exp.x.(n+1) = x ^ n * x$$
{Hipótesis Inductiva}
$$exp.x.(n+1) = exp.x.n * x$$

Anotemos el programa final y su modularización:
sum pot.x.0 = 0
sum pot.x.(n+1) = sum pot.x.n + exp.x.n
exp.x.0 = 1
exp.x.(n+1) = exp.x.n * x

- cubo.x = x^3

No nos queda otra opción que realizar inducción en “x”.

cubo.0 = 0 ^ 3
{Propiedad de potencias}
cubo.0 = 0

cubo.(x+1) = (x + 1)^3
{Propiedad de potencias}
cubo.(x+1) = x^3 + 3 * (x)^2 + 3 * x
{Hipótesis inductiva}
cubo.(x+1) = cubo.x + 3*x^2 + 3*x
{Modularizamos dos funciones}
cubo.(x+1) = cubo.x + triple(cuad.x) + triple.x

Veamos los módulos, definidos como:
- triple(cuad.x) = 3 * x^2
- triple.x = 3 * x

En el primer caso derivemos solo x^2.

cuad.0 = 0^2
{Propiedad de potencias}
cuad.0 = 0

cuad.(x+1) = (x + 1)^2
{Propiedad de potencias}
cuad.(x+1) = x^2 + 2x + 1
{Hipótesis Inductiva}
cuad.(x+1) = cuad.x + 2x + 1
{Modularizo}
cuad.(x+1) = cuad.x + doble.x + 1

Derivo doble.x = 2 * x

doble.0 = 2 * 0
{Propiedad multiplicación}
0

doble.(x+1) = 2 * (x+1)
{Distribución}
doble.(x+1) = 2 * x + 2
{Hipótesis inductiva}
doble.(x+1) = doble.x + 2

Ahora derivemos triple.x = 3*x

triple. 0 = 3 * 0
{Propiedad de multiplicación}
triple.0 = 0

triple.(x+1) = 3 * (x+1)
{Distribución}
triple.(x+1) = 3 *x + 3
{Hipótesis inductiva}
triple.(x+1) = triple.x + 3

Anotemos el programa final junto con sus módulos:
cubo.0 = 0
cubo.(x + 1) = cubo.x + triple(cuad.x) + triple.x
triple(cuad.0) = 0
triple(cuad.(x+1)) = 3 * (cuad.(x+1) = cuad.x + doble.x + 1)
doble.0 = 0
doble.(x+1) = doble.x + 2
triple.0 = 0
triple.(x+1) = triple.x + 3

- f.xs.ys = < ∀ i,j : 0 <= i < #xs ^ 0 <= j < #ys : xs.i ≠ ys.j >

Caso base con xs = [ ] // ys = [ ] o ambas listas vacías.

f.[ ].ys = < ∀ i,j : 0 <= i < #[ ] ^ 0 <= j < #ys : [ ].i ≠ ys.j >
{Cardinal de lista vacía}
f.[ ].ys = < ∀ i,j : 0 <= i < 0 ^ 0 <= j < #ys : [ ].i ≠ ys.j >
{Aritmética}
f.[ ].ys = < ∀ i,j : False ^ 0 <= j < #ys : [ ].i ≠ ys.j >
{Absorbente de la conjunción}
f.[ ].ys = < ∀ i,j : False : [ ].i ≠ ys.j >
{Rango vacío}
f.[ ].ys = True

$$f.xs.[\,] = \langle \forall\ i,j : 0 <= i < \#xs \wedge 0 <= j < \#[\,] : xs.i \neq [\,].j \rangle$$
{Cardinal de lista vacía}
$$f.xs.[\,] = \langle \forall\ i,j : 0 <= i < \#xs \wedge 0 <= j < 0: xs.i \neq [\,].j \rangle$$
{Aritmética}
$$f.xs.[\,] = \langle \forall\ i,j : 0 <= i < \#xs \wedge False: xs.i \neq [\,].j \rangle$$
{Absorbente de la conjunción}
$$f.xs.[\,] = \langle \forall\ i,j : False: xs.i \neq [\,].j \rangle$$
{Rango vacío}
$$f.xs.[\,] = True$$
$$f.[\,].[\,] = \langle \forall\ i,j : 0 <= i < \#[\,] \wedge 0 <= j < \#[\,] : [\,].i \neq [\,].j \rangle$$
{Propiedad de cardinal}
$$f.[\,].[\,] = \langle \forall\ i,j : 0 <= i < 0 \wedge 0 <= j < 0 : [\,].i \neq [\,].j \rangle$$
{Aritmética}
$$f.[\,].[\,] = \langle \forall\ i,j : False \wedge False : [\,].i \neq [\,].j \rangle$$
{Absorbente de la conjunción}
$$f.[\,].[\,] = \langle \forall\ i,j : False: [\,].i \neq [\,].j \rangle$$
{Rango vacío}
$$f.[\,].[\,] = True$$

Caso inductivo con xs = (x.xs) y ys = (y.ys)
$$f.(x.xs).(y.ys) = \langle \forall\ i,j : 0 <= i < \#(x.xs) \wedge 0 <= j < \#(y.ys) : (x.xs).i \neq (y.ys).j \rangle$$
{Anidado}
$$f.(x.xs).(y.ys) = \langle \forall\ i: 0 <= i < \#(x.xs) : \langle \forall\ j : 0 <= j < \#(y.ys) : (x.xs).i \neq (y.ys).j \rangle\rangle$$
{Abreviación}
$$f.(x.xs).(y.ys) = \langle \forall\ i: 0 <= i < \#(x.xs) : g.ys \rangle$$
{Aritmética en el rango}
$$f.(x.xs).(y.ys) = \langle \forall\ i: i = 0 \vee 1 <= i < \#(x.xs) : g.ys \rangle$$
{Partición de rango}
$$f.(x.xs).(y.ys) = \langle \forall\ i :1 <= i < \#(x.xs) : g.ys \rangle \wedge \langle \forall\ i : i = 0 : g.ys \rangle$$
{Cambio de variable en el primer término con i ← i + 1 y rango unitario}
$$f.(x.xs).(y.ys) = \langle \forall\ i :1 <= i + 1 < \#(x.xs) : \langle \forall\ j : 0 <= j < \#(y.ys) : (x.xs).i + 1 \neq (y.ys).j \rangle \rangle \wedge \langle \forall\ j : 0 <= j < \#(y.ys) : (x.xs).0 \neq (y.ys).j \rangle$$
{Propiedad de listas en ambas cuantificaciones}
$$f.(x.xs).(y.ys) = \langle \forall\ i :1 <= i + 1 < 1 + \#xs : \langle \forall\ j : 0 <= j < \#(y.ys) : xs.i \neq (y.ys).j \rangle \rangle \wedge \langle \forall\ j : 0 <= j < \#(y.ys) : x \neq (y.ys).j \rangle$$
{Aritmética en el rango de la primer cuantificación}
$$f.(x.xs).(y.ys) = \langle \forall\ i : 0 <= i < \#xs : \langle \forall\ j : 0 <= j < \#(y.ys) : xs.i \neq (y.ys).j \rangle \rangle \wedge \langle \forall\ j : 0 <= j < \#(y.ys) : x \neq (y.ys).j \rangle$$
{Desanidado}
$$f.(x.xs).(y.ys) = \langle \forall\ i,j : 0 <= i < \#xs \wedge 0 <= j < \#(y.ys) : xs.i \neq (y.ys).j \rangle \wedge \langle \forall\ j : 0 <= j < \#(y.ys) : x \neq (y.ys).j \rangle$$
{Aritmética en el rango de j}
$$f.(x.xs).(y.ys) = \langle \forall\ i,j : 0 <= i < \#xs \wedge (j = 0 \vee 1 <= j < \#(y.ys)) : xs.i \neq (y.ys).j \rangle \wedge \langle \forall\ j : 0 <= j < \#(y.ys) : x \neq (y.ys).j \rangle$$

{Distributividad y partición de rango}
f.(x.xs).(y.ys) = < ∀ i,j : 0 <= i < #xs ^ j = 0: xs.i ≠ (y.ys).j > ^
< ∀ i,j : 0 <= i <= #xs ^ 0 <= j < #(y.ys) : xs.i ≠ (y.ys).j > ^ < ∀ j : 0 <= j < #(y.ys) : x ≠ (y.ys).j >
{Eliminación de variable con j en el primer cuantificador}
f.(x.xs).(y.ys) = < ∀ i : 0 <= i < #xs: xs.i ≠ y > ^
< ∀ i,j : 0 <= i <= #xs ^ 1 <= j < #(y.ys) : xs.i ≠ (y.ys).j > ^ < ∀ j : 0 <= j < #(y.ys) : x ≠ (y.ys).j >
{Cambio de variable en el segundo cuantificador con j ← j + 1 y definición de cardinal}
f.(x.xs).(y.ys) = < ∀ i : 0 <= i < #xs: xs.i ≠ y > ^
< ∀ i,j : 0 <= i <= #xs ^ 1 <= j + 1 < 1 + #ys : xs.i ≠ (y.ys).j +1 > ^ < ∀ j : 0 <= j < #(y.ys) : x ≠ (y.ys).j >
{Aritmética en el rango y definición de indexación}
f.(x.xs).(y.ys) = < ∀ i : 0 <= i < #xs: xs.i ≠ y > ^
< ∀ i,j : 0 <= i <= #xs ^ 0 <= j < #ys : xs.i ≠ ys.j > ^ < ∀ j : 0 <= j < 1 + #ys : x ≠ (y.ys).j >
{Hipótesis inductiva}
f.(x.xs).(y.ys) = < ∀ i : 0 <= i < #xs: xs.i ≠ y > ^ f.xs.ys ^ < ∀ j : 0 <= j < 1 + #ys : x ≠ (y.ys).j >
{Modularizo el primer término}
mod1.y.xs ^ f.xs.ys ^ < ∀ j : 0 <= j < 1 + #ys : x ≠ (y.ys).j >
{Aritmética en el rango}
mod1.y.xs ^ f.xs.ys ^ < ∀ j : j = 0 v 1 <= j < 1 + #ys : x ≠ (y.ys).j >
{Partición de rango}
mod1.y.xs ^ f.xs.ys ^ < ∀ j :1 <= j < 1 + #ys : x ≠ (y.ys).j > ^ < ∀ j : j = 0: x ≠ (y.ys).j >
{Cambio de variable en el tercer término y rango unitario en el cuarto}
mod1.y.xs ^ f.xs.ys ^ < ∀ j :1 <= j + 1 < 1 + #ys : x ≠ (y.ys).j +1 > ^ (x ≠ y)
{Aritmética en el rango y definición de indexación}
mod1.y.xs ^ f.xs.ys ^ < ∀ j :<= j < #ys : x ≠ ys.j > ^ (x ≠ y)
{Modularizo de nuevo}
mod1.y.xs ^ f.xs.ys ^ mod2.x.ys ^ (x ≠ y)

mod1.y.xs = < ∀ i : 0 <= i < #xs: xs.i ≠ y >
Caso base con xs = [ ] y caso inductivo con xs = (x.xs)
mod1.y.[ ] = < ∀ i : 0 <= i < #[ ]: [ ].i ≠ x >
{Cardinal de lista vacía}
mod1.y.[ ] = < ∀ i : 0 <= i < 0 : [ ].i ≠ x >
{Aritmética}
mod1.y.[ ] = < ∀ i : False : [ ].i ≠ x >
{Rango vacío}
mod1.y.[ ] = True

mod1.y.(x.xs) = < ∀ i : 0 <= i < #(x.xs): (x.xs).i ≠ y >
{Aritmética en el rango y definición de cardinal}

$mod1.y.(x.xs) = \langle \forall i : i = 0 \vee 1 \leq i < 1 + \#xs : (x.xs).i \neq y \rangle$

{Partición de rango}

$mod1.y.(x.xs) = \langle \forall i : i = 0 \vee 1 \leq i < 1 + \#xs : (x.xs).i \neq y \rangle$

$mod1.y.(x.xs) = \langle \forall i : i = 0 : (x.xs).i \neq y \rangle \wedge \langle \forall i : 1 \leq i < 1 + \#xs : (x.xs).i \neq x \rangle$

{Rango unitario en el primer término y cambio de variable en el segundo $i \leftarrow i + 1$}

$mod1.y.(x.xs) = (x.xs).0 \neq y \wedge \langle \forall i : 1 \leq i + 1 < 1 + \#xs : (x.xs).i + 1 \neq x \rangle$

{Definición de indexación y aritmética en el rango}

$mod1.y.(x.xs) = (x \neq y) \wedge \langle \forall i : 0 \leq i < \#xs : xs.i \neq y \rangle$

{Hipótesis inductiva}

$mod1.y.(x.xs) = (x \neq y) \wedge mod1.xs$

$mod2.x.ys \langle \forall j : \leq j < \#ys : x \neq ys.j \rangle$

Caso base con ys = [ ] y caso inductivo con ys = (y.ys)

$mod2.x.ys \langle \forall j : \leq j < \#ys : x \neq ys.j \rangle$

{Analogo a mod1.xs}

…

True

$mod2.x.(y.ys) = \langle \forall j : \leq j < \#(y.ys) : x \neq (y.ys).j \rangle$

{Analogo a mod1.xs)

…

$mod2.x.(y.ys) = (x \neq y) \wedge mod2.ys$

Programa final:

f.[ ].ys = True

f.xs. [ ] = True

f.[ ].[ ] = True

$f.(x.xs).(y.ys) = mod1.xs \wedge f.xs.ys \wedge mod2.ys \wedge (x \neq y)$

mod1.y.[ ] = True

$mod1.y.(x.xs) = (x \neq y) \wedge mod1.xs$

mod2.x.[ ] = True

$mod2.x.(y.ys) = (x \neq y) \wedge mod2.ys$

- $f.xs = \langle \exists i, j : 0 \leq i < j < \#xs : xs.i = xs.j \rangle$

Derivamos mediante inducción, con xs = [ ] para el caso base y xs = (x.xs) para el caso inductivo.

$f.[\,] = \langle \exists i, j : 0 \leq i < j < \#[\,] : [\,].i = [\,].j \rangle$

{Definición de cardinal de lista}

$f.[\,] = \langle \exists i, j : 0 \leq i < j < 0 : [\,].i = [\,].j \rangle$

{Aritmética en el rango}

$f.[\,] = \langle \exists i, j : False : [\,].i = [\,].j \rangle$

{Rango vacío del existe}

f.[ ] = False

$$f.(x.xs) = < \exists\, i, j : 0 <= i < j < \#(x.xs) : (x.xs).i = (x.xs).j >$$

{Definición de cardinal de lista}

$$f.(x.xs) = < \exists\, i, j : 0 <= i < j < 1 + \#xs : (x.xs).i = (x.xs).j >$$

{Aritmética en el rango]

$$f.(x.xs) = < \exists\, i, j : (i = 0 \wedge < j < 1 + \#xs) \vee (0 < i \wedge j < 1 + \#xs) : (x.xs).i = (x.xs).j >$$

{Partición de rango}

{Aritmética en el rango]

$$f.(x.xs) = < \exists\, i, j : (0 < i \wedge j < 1 + \#xs) : (x.xs).i = (x.xs).j > \vee < \exists\, i, j : (i = 0 \wedge < j < 1 + \#xs) : (x.xs).i = (x.xs).j >$$

{Aritmética en el rango del primer cuantificador y eliminación de variable en el segundo}

$$f.(x.xs) = < \exists\, i, j : (1 <= i < j < 1 + \#xs) : (x.xs).i = (x.xs).j > \vee < \exists\, j : 0 < j < 1 + \#xs : (x.xs).0 = (x.xs).j >$$

{Cambios de variables en el primer término i ← i + 1, j ← j + 1}

$$f.(x.xs) = < \exists\, i, j : (1 <= i + 1 < j + 1 < 1 + \#xs) : (x.xs).i + 1 = (x.xs).j + 1 > \vee < \exists\, j : 0 < j < 1 + \#xs : (x.xs).0 = (x.xs).j >$$

{Aritmética en el rango, definición de indexación}

$$f.(x.xs) = < \exists\, i, j : 0 <= i < j< \#xs : xs.i = xs.j > \vee < \exists\, j : 0 < j < 1 + \#xs : x = (x.xs).j >$$

{Hipótesis inductiva}

$$f.(x.xs) = f.xs \vee < \exists\, j : 0 < j < 1 + \#xs : x = (x.xs).j >$$

{Modularización existe k}

$$f.(x.xs) = f.xs \vee existe.k.xs$$

Quedaría finalmente derivar existe.k.xs (lo hemos hecho varias veces y no presenta complicaciones).Programa final:

f. [ ] = False

f.(x.xs) = f.xs v existe.k.xs

- $g.xs.ys = < \exists\, as,cs :: xs = as ++ ys ++ ys ++ cs >$

Caso base con xs = [ ] y caso inductivo con xs = (x.xs)

$$g.(x.xs).ys = < \exists\, as,cs :: (x.xs) = as ++ ys ++ ys ++ cs >$$

{Tercero excluido}

$$g.(x.xs).ys = < \exists\, as,cs :: (as = [\,] \vee as \neq [\,]) \wedge (x.xs) = as ++ ys ++ ys ++ cs >$$

{Distributividad}

$$g.(x.xs).ys = < \exists\, as,cs :: (as = [\,] \wedge (x.xs) = as ++ ys ++ ys ++ cs) \vee (as \neq [\,] \wedge (x.xs) = as ++ ys ++ ys ++ cs) >$$

{Partición de rango}

$$g.(x.xs).ys = < \exists\, as,cs : as = [\,] : (x.xs) = as ++ ys ++ ys ++ cs > \vee < \exists\, as,cs : as \neq [\,] : (x.xs) = as ++ ys ++ ys ++ cs >$$

{Rango unitario en el primer término}

$$g.(x.xs).ys = < \exists\, cs : : (x.xs) = [\,] ++ ys ++ ys ++ cs > \vee < \exists\, as,cs : as \neq [\,] : (x.xs) = as ++ ys ++ ys ++ cs >$$

{Propiedad de listas en el primer término, reemplazo de as por (a.as) válido por tercero excluido}

$$g.(x.xs).ys = < \exists\, cs : : (x.xs) = ys ++ ys ++ cs > \vee < \exists\, as,cs : (a.as) \neq [\ ] : (x.xs) = (a.as) ++ ys ++ ys ++ cs >$$

{Aritmética en el rango de la segunda cuantificación}

$$g.(x.xs).ys = < \exists\, cs : : (x.xs) = ys ++ ys ++ cs > \vee < \exists\, a,as,cs : : (x.xs) = (a.as) ++ ys ++ ys ++ cs >$$

{Propiedad de ++}

$$g.(x.xs).ys = < \exists\, cs : : (x.xs) = ys ++ ys ++ cs > \vee < \exists\, a,as,cs : : x = a \wedge xs = as ++ ys ++ ys ++ cs >$$

$$g.(x.xs).ys = < \exists\, cs : : (x.xs) = ys ++ ys ++ cs > \vee < \exists\, a,as,cs : : x = a \wedge xs = as ++ ys ++ ys ++ cs >$$

{Distributividad}

$$g.(x.xs).ys = < \exists\, cs : : (x.xs) = ys ++ ys ++ cs > \vee (x = a \wedge < \exists\, as,cs : : xs = as ++ ys ++ ys ++ cs >)$$

{Hipótesis inductiva}

$$g.(x.xs).ys = < \exists\, cs : : (x.xs) = ys ++ ys ++ cs > \vee (x = a \wedge g.xs.ys)$$

{Modularización}

$$g.(x.xs).ys = mod1.cs \vee (x = a \wedge g.xs.ys)$$

- $h.xs.ys = < N\ p : 0 \leq p < \#xs : < \exists\ q : 0 \leq q < \#ys : xs\ !\ p = ys\ !\ q\ i >>$

Derivemos el caso base h. [ ]. ys

$$h.[\ ].ys = < N\ p : 0 \leq p < \#[\ ] : < \exists\ q : 0 \leq q < \#ys : [\ ]\ !\ p = ys\ !\ q\ i >>$$

{Definición de cardinal de lista vacía}

$$h.[\ ].ys = < N\ p : 0 \leq p < 0 : < \exists\ q : 0 \leq q < \#ys : [\ ]\ !\ p = ys\ !\ q\ i >>$$

{Aritmética en el rango}

$$h.[\ ].ys = < N\ p : False : < \exists\ q : 0 \leq q < \#ys : [\ ]\ !\ p = ys\ !\ q\ i >>$$

{Rango vacío}

$$h.[\ ].ys = 0$$

Derivemos el caso inductivo h.(x.xs).(y.ys)

$$h.(x.xs).(y.ys) = <N\ p : 0 \leq p < \#(x.xs) :< \exists\ q : 0 \leq q < \#(y.ys) : (x.xs)!\ p = (y.ys)!\ q >>$$

{Definición de cardinal de lista}

$$h.(x.xs).(y.ys) = <N\ p : 0 \leq p < 1 + \#xs : < \exists\ q : 0 \leq q < 1 + \#ys : (x.xs)!\ p = (y.ys)!\ q >>$$

{Aritmética en el primer rango}

$$h.(x.xs).(y.ys) = <N\ p : p = 0 \vee 1 \leq p < 1 + \#xs : < \exists\ q : 0 \leq q < 1 + \#ys : (x.xs)!\ p = (y.ys)!\ q >>$$

{Partición de rango}

h.(x.xs).(y.ys) = <N p : 1 ≤ p < 1 + #xs : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q >>
+ <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q >>
{Cambio de variable en el primer término, p ←p + 1}
h.(x.xs).(y.ys) = <N p : 1 ≤ p + 1 < 1 + #xs : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p + 1 = (y.ys)! q >> + <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q >>
{Aritmética en el rango del primer término, definición de indexación}
h.(x.xs).(y.ys) = <N p : 0 ≤ p < #xs : < ∃ q : 0 ≤ q < 1 + #ys : xs! p = (y.ys)! q >>
+ <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q >>
{Aritmética en el rango del existencial del primer término}
h.(x.xs).(y.ys) = <N p : 0 ≤ p < #xs : < ∃ q : q = 0 v 1 ≤ q < 1 + #ys : xs! p = (y.ys)! q >>
+ <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q >>
{Partición de rango sobre el mismo término}
h.(x.xs).(y.ys) = <N p : 0 ≤ p < #xs : < ∃ q : 1 ≤ q < 1 + #ys : xs! p = (y.ys)! q > v < ∃ q : q = 0 : xs! p = (y.ys)! q >>
+ <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q >>
{Cambio de variable sobre el mismo término q ← q + 1, rango unitario}
h.(x.xs).(y.ys) = <N p : 0 ≤ p < #xs : < ∃ q : 1 ≤ q + 1 < 1 + #ys : xs! p = (y.ys)! q + 1 > v (xs! p = (y.ys)! 0) >
+ <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q >>
{Aritmética en el rango y definición de indexación}
h.(x.xs).(y.ys) = <N p : 0 ≤ p < #xs: < ∃ q : 0 ≤ q < #ys : xs! p = ys! q > v (xs! p = y) >
+ <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q >>
{No puedo plantear hipótesis inductiva, por ende generalizo}
genh.k.(x.xs).(y.ys) = <N p : 0 ≤ p < #xs: < ∃ q : 0 ≤ q < #ys : xs! p = ys! q > v k >
+ <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q >>

Notamos que nuestra hipótesis inductiva era muy rígida y no nos permite seguir derivando, por eso propongo generalizar con la introducción de una nueva variable “k” que sea booleana.

genh.False.xs.ys = <N p : 0 ≤ p < #xs: < ∃ q : 0 ≤ q < #ys : xs! p = ys! q > v False >
{Neutro de la disyunción}
genh.False.xs.ys = <N p : 0 ≤ p < #xs: < ∃ q : 0 ≤ q < #ys : xs! p = ys! q > >
{Especificación de h.xs.ys}
genh.False.xs.ys = h.xs.ys

Ahora derivemos genh.False.xs.ys

genh.k.[ ].ys = <N p : 0 ≤ p < #[ ]: < ∃ q : 0 ≤ q < #ys : xs! p = ys! q > v k >
{Definición de cardinal de lista vacía}
genh.k.[ ].ys = <N p : 0 ≤ p < 0 : < ∃ q : 0 ≤ q < #ys : xs! p = ys! q > v k >
{Aritmética y rango vacío}
genh.k.[ ].ys = 0

genh.k.(x.xs).(x.xs) = <N p : 0 ≤ p < #(x.xs): < ∃ q : 0 ≤ q < #(y.ys) : (x.xs)! p = (y.ys)! q > v k >

{Definición de cardinal de lista}

genh.k.(x.xs).(x.xs) = <N p : 0 ≤ p < 1 + #xs: < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v k >

{Aritmética y partición de rango}

genh.k.(x.xs).(x.xs) = <N p : 1 ≤ p < 1 + #xs: < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v k > + <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v k >

{Cambio de variable p ← p + 1 y aritmética en el rango}

genh.k.(x.xs).(x.xs) = <N p : 0 ≤ p < #xs: < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p+1 = (y.ys)! q > v k > + <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v k >

{Definición de indexación}

genh.k.(x.xs).(x.xs) = <N p : 0 ≤ p < #xs: < ∃ q : 0 ≤ q < 1 + #ys : xs! p = (y.ys)! q > v k > + <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v k >

{Aritmética en el rango del existencial}

genh.k.(x.xs).(x.xs) = <N p : 0 ≤ p < #xs: < ∃ q : q = 0 v 1 <= q < 1 + #ys : xs! p = (y.ys)! q > v k > + <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v k >

{Partición de rango}

genh.k.(x.xs).(x.xs) = <N p : 0 ≤ p < #xs: < ∃ q : 1 <= q < 1 + #ys : xs! p = (y.ys)! q > v < ∃ q : q= 0 : xs! p = (y.ys)! q > v k > + <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v k >

{Cambio de variable q ← q + 1, aritmética en el rango, definición de indexación y rango unitario}

genh.k.(x.xs).(x.xs) = <N p : 0 ≤ p < #xs: < ∃ q : 0 <= q < #ys : xs! p = ys! q > v (xs! p = y) v k > + <N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v k >

{Hipótesis Inductiva}

genh.k.(x.xs).(x.xs) = genh.((xs! p = y) v k).xs.ys +
<N p : p = 0 : < ∃ q : 0 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v k >

{Aritmética en el rango del existencial y partición de rango}

genh.k.(x.xs).(x.xs) = genh.((xs! p = y) v k).xs.ys +
<N p : p = 0 : < ∃ q : 1 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v < ∃ q : q = 0: (x.xs)! p = (y.ys)! q > v k >

{Rango unitario}

genh.k.(x.xs).(x.xs) = genh.((xs! p = y) v k).xs.ys +
<N p : p = 0 : < ∃ q : 1 ≤ q < 1 + #ys : (x.xs)! p = (y.ys)! q > v (x.xs)! p = y > v k >

{Cambio de variable q ←q + 1 y aritmética en el rango}

genh.k.(x.xs).(x.xs) = genh.((xs! p = y) v k).xs.ys +
<N p : p = 0 : < ∃ q : 0 ≤ q < #ys : (x.x s)! p = ys! q > v (x.xs)! p = y > v k >

{Rango unitario}

(< ∃ q : 0 ≤ q < #ys : (x.xs)! 0 = ys! q > v (x.xs)! 0 = y > v k > → genh.((xs! p = y) v k).xs.ys + 1

¬ < ∃ q : 0 ≤ q < #ys : (x.xs)! 0 = ys! q > v (x.xs)! 0 = y > v k > →genh.((xs! p = y) v k).xs.ys

)

{Definición de índice}

(< ∃ q : 0 ≤ q < #ys : x = ys! q > v (x = y) > v k > → genh.((xs! p = y) v k).xs.ys + 1

¬ < ∃ q : 0 ≤ q < #ys : x = ys! q > v (x = y) > v k > →genh.((xs! p = y) v k).xs.ys

)

{Modularizo con existe.x.ys (trivial)}

(existe.x.ys v ((x = y) v k) → genh.((xs! p = y) v k).xs.ys + 1

¬ existe.x.ys v ((x = y) v k) →genh.((xs! p = y) v k).xs.ys

)

Programa final anotado:

h.[ ].ys = 0

h.xs.[ ] = 0

h.[ ].[ ] = 0

h.(x.xs).ys

h.xs.(y.ys)

h.(x.xs).(y.ys) = genh.False.xs.ys

genh.k.[ ].ys = 0

genh.k.(x.xs).(y.ys) = (existe.x.ys v ((x = y) v k) → genh.((xs! p = y) v k).xs.ys + 1

¬ existe.x.ys v ((x = y) v k) →genh.((xs! p = y) v k).xs.ys

)

- h.xs = < ∃ k : 0 <= k < #xs : sum.(xs ↑k) > k >

Derivamos mediante inducción en xs

h.[ ] = < ∃ k : 0 <= k < #[ ] : sum.(xs ↑k) > k >

{Definición de índice}

h.[ ] = < ∃ k : 0 <= k < 0 : sum.(xs ↑k) > k >

{Aritmética en el rango}

h.[ ] = < ∃ k : False : sum.(xs ↑k) > k >

{Rango vacío}

h.[ ] = False

h.(x.xs) = < ∃ k : 0 <= k < #(x.xs) : sum.(x.xs ↑k) > k >

{Definición de cardinal de lista}

h.(x.xs) = < ∃ k : 0 <= k < 1 + #xs : sum.((x.xs) ↑k) > k >

{Aritmética en el rango}

h.(x.xs) = < ∃ k : 0 = k v 1 <= k < 1 + #xs : sum.((x.xs) ↑k) > k >

{Partición de rango}

h.(x.xs) = < ∃ k :1 <= k < 1 + #xs : sum.((x.xs) ↑k) > k > v < ∃ k : k = 0 : sum.((x.xs) ↑k) > k >

{Cambio de variable k ←k + 1, y rango unitario}

h.(x.xs) = < ∃ k :1 <= k + 1 < 1 + #xs : sum.((x.xs) ↑k + 1) > k > v (sum.((x.xs) ↑0) > k + 1)

{Aritmética en el rango, definición de tomar)
h.(x.xs) = < ∃ k : 0 <= k <#xs : sum.(x.(xs↑k)) > k > v (sum.[ ] > k + 1)
{Definición de sum.[ ])
h.(x.xs) = < ∃ k :0 <= k < #xs : sum.(x.(xs↑k)) > k +1 > v (0 > k + 1)
{Definición de sum.(x.(xs↑k))}
h.(x.xs) = < ∃ k :0 <= k < #xs : x + sum.(xs↑k) > k + 1 > v (0 > k + 1)
{Mi hipótesis inductiva es demasiado rígida, propongo una generalización}
genh.t.x.xs = < ∃ k :0 <= k < #xs : x + sum.(xs↑k) > k + t >

Ahora probemos que genh.0.0.xs = h.xs
genh.0.xs = < ∃ k :0 <= k < #xs : 0 + sum.(xs↑k) > k >
{Elemento neutro de la suma}
genh.0.xs = < ∃ k :0 <= k < #xs : sum.(xs↑k) > k >
{Especificación de h.xs}
genh.0.0.xs = h.xs
Luego derivamos el caso base y el inductivo de nuestra nueva función generalizada

genh.x.[ ] = < ∃ k :0 <= k < #[ ] : x + sum.([ ]↑k) > k >
{Definición de indexación}
genh.x.[ ] = < ∃ k :0 <= k < 0 : x + sum.([ ]↑k) > k >
{Aritmética en el rango y rango vacío}
genh.x.[ ] = False

genh.t.x.(x.xs) = < ∃ k :0 <= k < #(x.xs) : x + sum.((x.xs)↑k) > k +t >
{Definición de cardinal de lista}
genh.t.x.(x.xs) = < ∃ k :0 <= k < 1 + #xs : x + sum.((x.xs)↑k) > k >
{Aritmética en el rango}
genh.t.x.(x.xs) = < ∃ k : k = 0 v 1 <= k < 1 + #xs : x + sum.((x.xs)↑k) > k >
{Partición de rango y rango unitario}
genh.t.x.(x.xs) = (x + sum.((x.xs)↑0) > k) v < ∃ k : 1 <= k < 1 + #xs : x + sum.((x.xs)↑k) > k + t >
{Cambio de variable k ← k + 1, aritmética en el rango}
genh.t.x.(x.xs) = (x + sum.((x.xs)↑0) > k) v < ∃ k : 0 <= k< #xs : x + sum.((x.xs)↑k+1) > k+ (t +1) >
{Definición de tomar}
genh.t.x.(x.xs) = (x + sum.[ ]> k) v < ∃ k : 0 <= k< #xs : x + sum.(x.(xs↑k)) > k + (t +1) >
{Definición de sum. [ ] y de sum.(x.(xs↑k))}
genh.t.x.(x.xs) = (x + 0 > k) v < ∃ k : 0 <= k < #xs : (x + x) + sum.(xs↑k) > k + (t +1) >
{Hipótesis inductiva}
genh.t.x.(x.xs) = (x + 0 > k) v genh.(t + 1).(x + x)

Programa final anotado:
h.[ ] = False

h.(x.xs) = hgen.0.0.xs
hgen.t.x.[ ] = False
hgen.t.x.(x.xs) = (x + 0 > k) v genh.(t + 1).(x + x)

- f.xs.ys = < Max as, bs, cs : xs = as ++ bs ^ ys = as ++ cs : prod.as >

1. Explicar con tus palabras que calcula f.
2. Derivar el caso inductivo hasta llegar a generalizar.
3. Especifica la función generalizada y proba que fgen es un caso particular de f.
4. Deriva la función generalizada.

1. La función f calcula la máxima multiplicación de un segmento inicial entre dos listas.
2. Hagamos inducción en xs y análisis por casos en ys:

< Max as, bs, cs : xs = as ++ bs ^ ys = as ++ cs : prod.as >
{Especificación}
< Max as, bs, cs : (x.xs) = as ++ bs ^ ys = as ++ cs : prod.as >
{Tercero excluido con as = [ ] v as ≠ [ ]}
< Max as, bs, cs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ bs ^ ys = as ++ cs : prod.as >
{Distributividad}
< Max as, bs, cs : (as = [ ] ^ (x.xs) = as ++ bs ^ ys = as ++ cs) v (as ≠ [ ] ^ (x.xs) = as ++ bs ^ ys = as ++ cs) : prod.as >
{Partición de rango}
< Max as, bs, cs : (as = [ ] ^ (x.xs) = as ++ bs ^ ys = as ++ cs) : prod.as > max
< Max as, bs, cs : (as ≠ [ ] ^ (x.xs) = as ++ bs ^ ys = as ++ cs): prod.as >
{Eliminación de variable en el primer término}
< Max bs, cs : (x.xs) = [ ] ++ bs ^ ys = [ ] ++ cs : prod.[ ] > max
< Max as, bs, cs : (as ≠ [ ] ^ (x.xs) = as ++ bs ^ ys = as ++ cs): prod.as >
{Propiedad de listas}
< Max bs, cs : (x.xs) = bs ^ ys = cs : prod.[ ] > max
< Max as, bs, cs : (as ≠ [ ] ^ (x.xs) = as ++ bs ^ ys = as ++ cs): prod.as >
{Eliminación de variable con bs e ys}
prod.[ ] max
< Max as, bs, cs : (as ≠ [ ] ^ (x.xs) = as ++ bs ^ ys = as ++ cs): prod.as >
{Reemplazo de as por (a.as), válido por tercero excluido}
prod.[ ] max
< Max a,as, bs, cs : (x.xs) = (a.as) ++ bs ^ ys = (a.as) ++ cs: prod.(a.as) >
{Propiedad de listas}
prod.[ ] max < Max a,as, bs, cs : (x = a) ^ xs = as ++ bs ^ ys = (a.as) ++ cs: prod.(a.as) >
{Definición de prod}
prod.[ ] max < Max a,as, bs, cs : (x = a) ^ xs = as ++ bs ^ ys = (a.as) ++ cs : a * prod.as >
{Eliminación de variable con x = a}

prod.[ ] max < Max as, bs, cs : xs = as ++ bs ^ ys = (a.as) ++ cs : x * prod.as >
{Análisis por casos, ys = [ ] v ys ≠ [ ])

Caso ys = [ ] :

prod.[ ] max < Max as, bs, cs : xs = as ++ bs ^ [ ] = (a.as) ++ cs : x * prod.as >
{Propiedad de listas}
prod.[ ] max < Max as, bs, cs : xs = as ++ bs ^ False : x * prod.as >
{Absorbente de la conjunción}
prod.[ ] max < Max as, bs, cs : False : x * prod.as >
{Rango vacío del cuantificador máximo}
prod.[ ] max -inf
{Aritmética}
prod.[ ]

Caso ys = (y.ys)

prod.[ ] max < Max y,as, bs, cs : xs = as ++ bs ^ (y.ys) = (a.as) ++ cs : x * prod.as >
{Propiedad de listas}
prod.[ ] max < Max y,as, bs, cs : xs = as ++ bs ^ (y = a) ^ ys = as ++ cs : x * prod.as >
{Eliminación de variable con y = a}
prod.[ ] max < Max as, bs, cs : xs = as ++ bs ^ ys = as ++ cs : x * prod.as >
{Es necesario generalizar, no puedo aplicar hipótesis inductiva}
genf.xs.ys.k = < Max as, bs, cs : xs = as ++ bs ^ ys = as ++ cs : k * prod.as >

3. genf.xs.ys.k = < Max as, bs, cs : xs = as ++ bs ^ ys = as ++ cs : k * prod.as >
Luego probemos que es un caso particular de f.xs.ys:

< Max as, bs, cs : xs = as ++ bs ^ ys = as ++ cs : k * prod.as >
{Elijo k ← 1}
< Max as, bs, cs : xs = as ++ bs ^ ys = as ++ cs : 1 * prod.as >
{Neutro del producto}
< Max as, bs, cs : xs = as ++ bs ^ ys = as ++ cs : prod.as >
{Definición de f.xs.ys}
f.xs.ys

Listo, queda probado que genf.xs.ys.1 = f.xs.ys

4. Derivemos ahora la función generalizada:

Para el caso base tomemos xs = [ ]

< Max as, bs, cs : [ ] = as ++ bs ^ ys = as ++ cs : k * prod.as >
{Propiedad de listas}
< Max as, bs, cs : (as = [ ] ^ bs = [ ]) ^ ys = as ++ cs : k * prod.as >

{Eliminación de variable con as = [ ] y bs = [ ]}
< Max cs : ys = [ ] ++ cs : k * prod.[ ] >
{Propiedad de listas}
< Max cs : ys = cs : k * prod.[ ] >
{Rango unitario}
k * prod.[ ]

Para el caso inductivo tomemos xs = (x.xs) y hagamos análisis por casos en ys.

< Max as, bs, cs : (x.xs) = as ++ bs ^ ys = as ++ cs : k * prod.as >
(Tercero excluido)
< Max as, bs, cs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ bs ^ ys = as ++ cs : k * prod.as >
(Distributividad y partición de rango}
< Max as, bs, cs : as = [ ] ^ (x.xs) = as ++ bs ^ ys = as ++ cs : k * prod.as > max < Max as, bs, cs : as ≠ [ ] ^ (x.xs) = as ++ bs ^ ys = as ++ cs : k * prod.as >
(Eliminación de variable en el primer término, reemplazo de as por (a.as) válido por tercero excluido)
< Max a, bs, cs : (x.xs) = [ ] ++ bs ^ ys = [ ] ++ cs : k * prod. [ ] > max
< Max a, as, bs, cs : (x.xs) = (a.as) ++ bs ^ ys = (a.as) ++ cs : k * prod.(a.as) >
{Propiedad de listas en ambos términos}
< Max bs, cs : (x.xs) = bs ^ ys = cs : k * prod. [ ] > max
< Max a, as, bs, cs : (x = a) ^ xs = as ++ bs ^ ys = (a.as) ++ cs : k * prod.(a.as) >
{Definición de prod y eliminación de variable con x = a}
< Max bs, cs : (x.xs) = bs ^ ys = cs : k * prod. [ ] > max
< Max as, bs, cs : xs = as ++ bs ^ ys = (a.as) ++ cs : (k * x) * prod.as >
{Eliminación de variable con bs y cs en el primer término}
k * prod. [ ] max
< Max as, bs, cs : xs = as ++ bs ^ ys = (a.as) ++ cs : (k * x) * prod.as >

Caso ys = [ ]

k * prod. [ ] max
< Max as, bs, cs : xs = as ++ bs ^ [ ] = (a.as) ++ cs : (k * x) * prod.as >
{Aritmética en el rango}
k * prod. [ ] max
< Max as, bs, cs : xs = as ++ bs ^ False : (k * x) * prod.as >
{Absorbente de la conjunción y rango vacío}
k * prod. [ ] max -inf
{Aritmética}
k * prod. [ ]

Caso ys = (y.ys)

k * prod. [ ] max
< Max y, as, bs, cs : xs = as ++ bs ^ (y.ys) = (a.as) ++ cs : (k * x) * prod.as >

{Propiedad de listas}
k * prod. [ ] max
< Max y, as, bs, cs : xs = as ++ bs ^ (y = a) ^ ys = as ++ cs : (k * x) * prod.as >
{Eliminación de variable con y = a}
k * prod. [ ] max
< Max y, as, bs, cs : xs = as ++ bs ^ ys = as ++ cs : (k * x) * prod.as >
{Hipótesis inductiva}
k * prod. [ ] max genf.xs.ys.(k * x)

Programa final, anotado:
f.[ ].ys = prod.[ ]
f.(x.xs).[ ] = prod.[ ]
f.(x.xs).(y.ys) = prod. [ ] max genf.xs.ys.1
genf.[ ].ys = k * prod.[ ]
genf.(x.xs).[ ] = k * prod. [ ]
genf.(x.xs).(y.ys) = k * prod. [ ] max genf.xs.ys.(k * x)

- h.xs = < ∃ as, bs : xs = as ++ bs : 2 ∗ sum.as = #as + 1 >

Para derivar este programa, hagamos inducción en xs con el caso base xs = [ ] y el caso inductivo xs = (x.xs).

h.[ ] = < ∃ as, bs : [ ] = as ++ bs : 2 ∗ sum.as = #as + 1 >
{Propiedad de listas}
h.[ ] = < ∃ as, bs : [ ] = as ^ bs = [ ] : 2 ∗ sum.as = #as + 1 >
{Anidado}
h.[ ] = < ∃ bs :bs = [ ] : < ∃ as : as = [ ] : 2 ∗ sum.as = #as + 1 >>
{Rango unitario}
< ∃ as : as = [ ] : 2 ∗ sum.as = #as + 1 >
{Rango unitario}
2 ∗ sum.[ ] = #[ ] + 1
{Aritmética, definición de sum.[ ] y cardinal de lista vacía}
0 = 1
{Aritmética}
False

h.(x.xs) = < ∃ as, bs : (x.xs) = as ++ bs : 2 ∗ sum.as = #as + 1 >
{Tercero excluido con as = [ ] o as ≠ [ ]}
h.(x.xs) = < ∃ as, bs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ bs : 2 ∗ sum.as = #as + 1 >
{Distributividad}
h.(x.xs) = < ∃ as, bs : (as = [ ] ^ (x.xs) = as ++ bs) v (as ≠ [ ] ^ (x.xs) = as ++ bs) : 2 ∗ sum.as = #as + 1 >
{Partición de rango}

h.(x.xs) = < ∃ as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs : 2 ∗ sum.as = #as + 1 > v

< ∃ as, bs : as = [ ] ^ (x.xs) = as ++ bs : 2 ∗ sum.as = #as + 1 >

{Eliminación de variable con as = [ ] en el segundo cuantificador}

h.(x.xs) = < ∃ as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs : 2 ∗ sum.as = #as + 1 > v

< ∃ bs : (x.xs) = [ ] ++ bs : 2 ∗ sum.[ ] = #[ ] + 1 >

{Propiedad de listas, definición de sum.[ ] y cardinal de listas en el segundo cuantificador}

h.(x.xs) = < ∃ as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs : 2 ∗ sum.as = #as + 1 > v

< ∃ bs : (x.xs) = bs : 2 ∗ 0 = 0 + 1 >

{Rango unitario}

h.(x.xs) = < ∃ as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs : 2 ∗ sum.as = #as + 1 > v

(2 ∗ 0 = 0 + 1)

{Aritmética}

h.(x.xs) = < ∃ as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs : 2 ∗ sum.as = #as + 1 > v

(0 = 1)

{Lógica y elemento neutro de la disyunción}

h.(x.xs) = < ∃ as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs : 2 ∗ sum.as = #as + 1 >

{Reemplazo de as por (a.as) válido por tercero excluido}

h.(x.xs) = < ∃ a,as, bs : (a.as) ≠ [ ] ^ (x.xs) = (a.as) ++ bs : 2 ∗ sum.(a.as) = #(a.as) + 1 >

Aritmética, propiedad de listas, definición de sum.(a.as) y cardinal de listas}

h.(x.xs) = < ∃ a,as, bs : True ^ (x = a) ^ xs = as ++ bs : 2 ∗ (a + sum.as) = (1 + #as) + 1 >

{Elemento neutro de la conjunción y eliminación de variable con x = a}

h.(x.xs) = < ∃ as, bs : xs = as ++ bs : 2 ∗ (x + sum.as) = (1 + #as) + 1 >

{Aritmética en el término}

h.(x.xs) = < ∃ as, bs : xs = as ++ bs : 2 ∗ (x + sum.as) = 2 + #as >

{No puedo aplicar hipótesis inductiva, propongo generalización}

genh.xs.q.r = < ∃ as, bs : xs = as ++ bs : 2 ∗ (q + sum.as) = r + #as >

Veamos ahora qué genh.xs.q.r es un caso particular de h.xs:

genh.xs.0.1 = < ∃ as, bs : xs = as ++ bs : 2 ∗ (0 + sum.as) = 1 + #as >

{Elemento neutro de la suma}

genh.xs.0.1 = < ∃ as, bs : xs = as ++ bs : 2 ∗ sum.as = 1 + #as >

{Especificación de h.xs}

genh.xs.0.1 = h.xs

Derivemos genh.xs.q.r = < ∃ as, bs : xs = as ++ bs : 2 ∗ (q + sum.as) = r + #as >.

genh.[ ].q.r = < ∃ as, bs : [ ] = as ++ bs : 2 ∗ (q + sum.as) = r + #as >

{Propiedad de listas}

genh.[ ].q.r = < ∃ as, bs : [ ] = as ^ bs = [ ] : 2 ∗ (q + sum.as) = r + #as >

{Eliminación de variable bs = [ ] y rango unitario}

$$2 * (q + sum.[\ ]) = r + \#[\ ]$$
{Definición de sum.[ ] y cardinal de listas}
$$2 * (q + 0) = r + 0$$
{Aritmética}
$$(2 * q) = r$$

$$genh.(x.xs).q.r = \langle \exists\ as, bs : (x.xs) = as +\!\!+ bs : 2 * (q + sum.as) = r + \#as \rangle$$
{Tercero excluido con as = [ ] o as ≠ [ ]}
$$genh.(x.xs).q.r = \langle \exists\ as, bs : (as = [\ ] \lor as \neq [\ ]) \land (x.xs) = as +\!\!+ bs : 2 * (q + sum.as) = r + \#as \rangle$$
{Distributividad}
$$genh.(x.xs).q.r = \langle \exists\ as, bs : (as = [\ ] \land (x.xs) = as +\!\!+ bs) \lor (as \neq [\ ] \land (x.xs) = as +\!\!+ bs) : 2 * (q + sum.as) = r + \#as \rangle$$
{Partición de rango}
$$genh.(x.xs).q.r = \langle \exists\ as, bs : as \neq [\ ] \land (x.xs) = as +\!\!+ bs : 2 * (q + sum.as) = r + \#as \rangle \lor \langle \exists\ as, bs : as = [\ ] \land (x.xs) = as +\!\!+ bs : 2 * (q + sum.as) = r + \#as \rangle$$
{Reemplazo de as ← (a.as) y eliminación de variable con as = [ ]}
$$genh.(x.xs).q.r = \langle \exists\ a,as, bs : (a.as) \neq [\ ] \land (x.xs) = (a.as) +\!\!+ bs : 2 * (q + sum.(a.as)) = r + \#(a.as) \rangle \lor \langle \exists\ as, bs : (x.xs) = [\ ] +\!\!+ bs : 2 * (q + sum.[\ ]) = r + \#[\ ] \rangle$$
{Aritmética, propiedad de listas, definición de sum y de cardinal de listas}
$$genh.(x.xs).q.r = \langle \exists\ a,as, bs : True \land (x = a) \land xs = as +\!\!+ bs : 2 * ((q + a) + sum.as) = r + 1 + \#as \rangle \lor \langle \exists\ as, bs : (x.xs) = bs : 2 * (q + 0) = r + 0 \rangle$$
{Neutro de la conjunción, eliminación de variable con x = a, aritmética y rango unitario}
$$genh.(x.xs).q.r = \langle \exists\ as, bs : xs = as +\!\!+ bs : 2 * ((q + x) + sum.as) = (r + 1) + \#as \rangle \lor (2 * q) = r$$
{Hipótesis inductiva}
$$genh.(x.xs).q.r = genh.xs.(q + x).(r + 1) \lor (2 * q) = r$$

Programa final, anotado:
h.[ ] = False
h.(x.xs) = genh.xs.0.1
genh.[ ].q.r = ((2 ∗ q) = r)
genh.xs.q.r = genh.xs.(q + x).(r + 1) v ((2 ∗ q) = r)

- $f.xs.n = \langle \exists\ as, bs : xs = as +\!\!+ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\ !i * (n - i) \rangle = 8 \rangle$

Derivemos f.xs.n mediante inducción en xs.

$$\langle \exists\ as, bs : xs = as +\!\!+ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\ !i * (n - i) \rangle = 8 \rangle$$
{Caso base con xs [ ]}
$$\langle \exists\ as, bs : [\ ] = as +\!\!+ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\ !i * (n - i) \rangle = 8 \rangle$$

{Propiedad de listas}

$$< \exists\ as, bs : [\ ] = as \wedge bs = [\ ] : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Anidado}

$$< \exists\ as : [\ ] = as : < \exists\ bs = [\ ] : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 > >$$

{Rango unitario}

$$< \exists\ bs = [\ ] : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Rango unitario}

$$< \Sigma\, i : 0 \leq i < \#[\ ] : [\ ]\,!i \ast (n - i) > = 8 >$$

{Definición de cardinal de lista}

$$< \Sigma\, i : 0 \leq i < 0 : [\ ]\,!i \ast (n - i) > = 8 >$$

{Aritmética y rango vacío}

$$0$$

$$< \exists\ as, bs : xs = as \mathbin{++} bs : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Caso inductivo con xs = (x.xs)

$$< \exists\ as, bs : (x.xs) = as \mathbin{++} bs : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Tercero excluido con as = [ ] v as ≠ [ ]}

$$< \exists\ as, bs : (x.xs) = as \mathbin{++} bs \wedge (as = [\ ] \vee as \neq [\ ]) : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Distributividad}

$$< \exists\ as, bs : ((x.xs) = as \mathbin{++} bs \wedge as = [\ ]) \vee ((x.xs) = as \mathbin{++} bs \wedge as \neq [\ ]) : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Partición de rango}

$$< \exists\ as, bs : (x.xs) = as \mathbin{++} bs \wedge as = [\ ] : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 > \vee$$
$$< \exists\ as, bs : (x.xs) = as \mathbin{++} bs \wedge as \neq [\ ] : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Eliminación de variable en el primer término, reemplazo de as ← a.as}

$$< \exists\ bs : (x.xs) = [\ ] \mathbin{++} bs : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 > \vee$$
$$< \exists\ a,as, bs : (x.xs) = (a.as) \mathbin{++} bs \wedge (a.as) \neq [\ ] : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Propiedad de listas en el primer término, y en el segundo}

$$< \exists\ bs : (x.xs) = bs : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 > \vee$$
$$< \exists\ a,as, bs : (x.xs) = (a.as) \mathbin{++} bs \wedge True : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Rango unitario en el primer término, neutro de la conjunción en el segundo}

$$< \Sigma\, i : 0 \leq i < \#(x.xs) : (x.xs)\,!i \ast (n - i) > = 8 \ \vee$$
$$< \exists\ a,as, bs : (x.xs) = (a.as) \mathbin{++} bs : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Definición de cardinal en el primer término, propiedad de listas en el segundo}

$$< \Sigma\, i : 0 \leq i < 1 + \#xs : (x.xs)\,!i \ast (n - i) > = 8 \ \vee$$
$$< \exists\ a,as, bs : (x.xs) = a.(as \mathbin{++} bs) : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Propiedad de listas en el segundo término}

$$< \Sigma\, i : 0 \leq i < 1 + \#xs : (x.xs)\,!i \ast (n - i) > = 8 \ \vee$$
$$< \exists\ a,as, bs : (x = a) \wedge xs = as \mathbin{++} bs : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Eliminación de variable con a = x}

$$< \Sigma\, i : 0 \leq i < 1 + \#xs : (x.xs)\,!i \ast (n - i) > = 8 \ \vee$$
$$< \exists\ as, bs : xs = as \mathbin{++} bs : < \Sigma\, i : 0 \leq i < \#bs : bs\,!i \ast (n - i) > = 8 >$$

{Hipótesis inductiva, sigo trabajando el primer término}

$< \Sigma\ i : 0 \leq i < 1 + \#xs : (x.xs)\ !i \ast (n - i) > \ = 8\ \ v\ f.xs.n$

{Aritmética en el rango}

$< \Sigma\ i : i = 0\ v\ 1 \leq i < 1 + \#xs : (x.xs)\ !i \ast (n - i) > \ = 8\ \ v\ f.xs.n$

{Partición de rango}

$< \Sigma\ i : 1 \leq i < 1 + \#xs : (x.xs)\ !i \ast (n - i) > v < \Sigma\ i : i = 0 : (x.xs)\ !i \ast (n - i) > \ = 8\ \ v\ f.xs.n$

{Cambio de variable $i \leftarrow i + 1$, rango unitario}

$< \Sigma\ i : 1 \leq i + 1 < 1 + \#xs : (x.xs)\ !i + 1 \ast (n - (i + 1)) > v\ ((x.xs)\ !0 \ast (n - 0))\ \ = 8\ \ v\ f.xs.n$

{Aritmética en el rango, definición de indexación, y suma}

$< \Sigma\ i : 0 \leq i < \#xs : xs\ !i \ast (n - (i + 1)) > v\ (x \ast n\ )\ \ = 8\ \ v\ f.xs.n$

{Modularización del primer término}

$(mod1.n.xs\ v\ (x \ast n\ )\ \ = 8)\ \ v\ f.xs.n$

Derivemos mod1.n.xs mediante inducción en xs.

$< \Sigma\ i : 0 \leq i < \#xs : xs\ !i \ast (n - (i + 1)) >$

{Caso base con xs = [ ]}

$< \Sigma\ i : 0 \leq i < \#[\ ] : [\ ]\ !i \ast (n - (i + 1)) >$

{Definición de cardinal, aritmética y rango vacío}

0

$< \Sigma\ i : 0 \leq i < \#xs : xs\ !i \ast (n - (i + 1)) >$

{Caso inductivo con xs= (x.xs)}

$< \Sigma\ i : 0 \leq i < \#(x.xs) : (x.xs)\ !i \ast (n - (i + 1)) >$

{Definición de cardinal y aritmética en el rango}

$< \Sigma\ i : i = 0\ v\ 1 \leq i < 1 + \#xs : (x.xs)\ !i \ast (n - (i + 1)) >$

{Partición de rango}

$< \Sigma\ i : 1 \leq i < 1 + \#xs : (x.xs)\ !i \ast (n - (i + 1)) > +$

$< \Sigma\ i : i = 0 : (x.xs)\ !i \ast (n - (i + 1)) >$

{Cambio de variable $i \leftarrow i + 1$, rango unitario}

$< \Sigma\ i : 1 \leq i + 1 < 1 + \#xs : (x.xs)\ !i + 1 \ast (n - ((i + 1) + 1)) > +$

$(x.xs)\ !0 \ast (n - (0 + 1))$

{Aritmética en el rango, definición de indexación y propiedades de suma}

$< \Sigma\ i : 0 \leq i\ \ < \#xs : xs\ !i\ \ \ast (n - ((i + 1) + 1)) > +$

$x \ast (n - 1)$

{Aritmética}

$< \Sigma\ i : 0 \leq i\ \ < \#xs : xs\ !i\ \ \ast (n - (i + 1) + 1)) > + x \ast (n - 1)$

{No puedo aplicar hipótesis inductiva, debo generalizar}

$genmod1.n.p.xs = < \Sigma\ i : 0 \leq i\ \ < \#xs : xs\ !i\ \ \ast (n - (i + p) + 1) >$

$genmod1.n.0.xs$

{Especificación}

$< \Sigma\ i : 0 \leq i\ \ < \#xs : xs\ !i\ \ \ast (n - ((i + 0) + 1)) >$

{Elemento neutro de la suma}

$$< \Sigma\, i : 0 \le i \ < \#xs : xs\ !i\ * (n - (i\ + 1)) >$$
$$\{\text{Especificación de mod1.n.xs}\}$$
$$mod1.n.xs$$

Ya está probado que genmod1.n.p.xs es un caso particular de mod1.n.xs, ahora derivemos la función generalizada.

$$< \Sigma\, i : 0 \le i \ < \#xs : xs\ !i\ * (n - ((i + p) + 1)) >$$
$$\{\text{Caso base con } xs = [\ ]\}$$
$$< \Sigma\, i : 0 \le i \ < \#[\ ] : [\ ]\ !i\ * (n - ((i + p) + 1)) >$$
$$\ldots$$
$$\{\text{Rango vacío}\}$$
$$0$$

$$< \Sigma\, i : 0 \le i \ < \#xs : xs\ !i\ * (n - ((i + p) + 1)) >$$
$$\{\text{Caso inductivo con } xs = (x.xs)\}$$
$$< \Sigma\, i : 0 \le i \ < \#(x.xs) : (x.xs)\ !i\ * (n - ((i + p) + 1)) >$$
$$\{\text{Definición de cardinal de lista, aritmética en el rango}\}$$
$$< \Sigma\, i : i = 0 \lor 1 \le i \ < 1 + \#xs : (x.xs)\ !i\ * (n - ((i + p) + 1)) >$$
$$\{\text{Partición de rango, rango unitario y cambio de variable } i \leftarrow i + 1\}$$
$$< \Sigma\, i : 1 \le i + 1 < 1 + \#xs : (x.xs)\ !i + 1\ * (n - (((i + 1) + p) + 1)) > \lor\ (x * (\ n - (p + 1)\ )\ )$$
$$\{\text{Aritmética en el rango, definición de indexación}\}$$
$$< \Sigma\, i : 0 \le i < \#xs : xs\ !i\ * (n - (((i + 1) + p) + 1)) > \lor\ (x * (\ n - (p + 1)\ )\ )$$
$$\{\text{Aritmética}\}$$
$$< \Sigma\, i : 0 \le i < \#xs : xs\ !i\ * (n - ((i + (1 + p)) + 1)) > \lor\ (x * (\ n - (p + 1)\ )\ )$$
$$\{\text{Hipótesis inductiva}\}$$
$$genmod.n.(1 + p).xs \lor (x * (\ n - (p + 1)\ )\ )$$

Programa final,anotado:
f.n.[ ] = 0
f.n.(x.xs) = (mod1.n.xs v (x ∗ n) = 8) v f.xs.n
mod1.n.[ ] = 0
mod1.n.(x.xs) = genmod1.n.0.xs + (x ∗ (n − 1))
genmod1.n.p.[ ] = 0
genmod1.n.p.(x.xs) = genmod.n.(1 + p).xs v (x ∗ (n − (p + 1)))

- f.xs = < Max as, bs, cs : xs = as ++ bs ++ cs ∧ as = cs : #as >

Ayuda: para generalizar considera que x : as = [x] ++ as.

$$f.[\ ] = < Max\ as, bs, cs : [\ ] = as +\!+ bs +\!+ cs \land as = cs : \#as >$$
$$\{\text{Propiedad de listas}\}$$
$$f.[\ ] = < Max\ as, bs, cs : as = [\ ] \wedge bs = [\ ] \wedge cs = [\ ] \land as = cs : \#as >$$
$$\{\text{Eliminación de variable con } as = [\ ]\}$$
$$f.[\ ] = < Max\ as, bs, cs : bs = [\ ] \wedge cs = [\ ] \land [\ ] = cs : \#[\ ] >$$

{Eliminación de cs = [ ]}

f.[ ] = < Max as, bs, cs : bs = [ ] ∧ [ ] = [ ] : #[ ] >

{Aritmética}

f.[ ] = < Max as, bs, cs : bs = [ ] ∧ True : #[ ] >

{Neutro de la conjunción, y rango unitario}

f. [ ] = #[ ]

{Definición de cardinal}

f. [ ] = 0

f.(x.xs) = < Max as, bs, cs : (x.xs) = as ++ bs ++ cs ∧ as = cs : #as >

{Tercero excluido}

f.(x.xs) = < Max as, bs, cs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ bs ++ cs ∧ as = cs : #as >

{Distributividad}

f.(x.xs) = < Max as, bs, cs : as = [ ] ^ (x.xs) = as ++ bs ++ cs ∧ as = cs : #as > max
< Max as, bs, cs : as ≠ [ ] ^ (x.xs) = as ++ bs ++ cs ∧ as = cs : #as >

{Eliminación de variable en el primer término con as = [ ], cambio de variable en el segundo término as ← (a.as)}

f.(x.xs) = < Max bs, cs : (x.xs) = [ ] ++ bs ++ cs ∧ [ ] = cs : #[ ] > max
< Max a,as, bs, cs : (a.as) ≠ [ ] ^ (x.xs) = (a.as) ++ bs ++ cs ∧ (a.as) = cs : #(a.as) >

{Propiedad de listas}

f.(x.xs) = < Max bs, cs : (x.xs) = bs ++ cs ∧ [ ] = cs : #[ ] > max
< Max a,as, bs, cs : True ^ (x.xs) = (a.as) ++ bs ++ cs ∧ (a.as) = cs : #(a.as) >

{Eliminación de variable en el primer término con cs = [ ]}

f.(x.xs) = < Max bs : (x.xs) = bs ++ [ ] : #[ ] > max
< Max a,as, bs, cs : True ^ (x.xs) = (a.as) ++ bs ++ cs ∧ (a.as) = cs : #(a.as) >

{Propiedad de listas}

f.(x.xs) = < Max bs : (x.xs) = bs : #[ ] > max
< Max a,as, bs, cs : True ^ (x.xs) = (a.as) ++ bs ++ cs ∧ (a.as) = cs : #(a.as) >

{Rango unitario en el primer término}

#[ ] max < Max a,as, bs, cs : True ^ (x.xs) = (a.as) ++ bs ++ cs ∧ (a.as) = cs : #(a.as) >

{Cardinal de lista}

0 max < Max a,as, bs, cs : True ^ (x.xs) = (a.as) ++ bs ++ cs ∧ (a.as) = cs : #(a.as) >

{Neutro de la conjunción, cardinal de lista}

0 max < Max a,as, bs, cs : (x.xs) = (a.as) ++ bs ++ cs ∧ (a.as) = cs : 1 + #as >

{Propiedad de listas}

0 max < Max a,as, bs, cs : (x = a) ^ xs = as ++ bs ++ cs ∧ (a.as) = cs : 1 + #as >

{Eliminación de variable con x = a}

0 max < Max as, bs, cs : xs = as ++ bs ++ cs ∧ (x.as) = cs : 1 + #as >

{Usamos la ayuda : (x.as) = [x] ++ as}

0 max < Max as, bs, cs : xs = as ++ bs ++ cs ∧ ([x] ++ as) = cs : 1 + #as >

{Generalizamos}

genf.ks.z.xs = < Max as, bs, cs : xs = as ++ bs ++ cs ∧ ks ++ as = cs : z + #as >

genf.[ ].0.xs = < Max as, bs, cs : xs = as ++ bs ++ cs ∧ [ ] ++ as = cs : 0 + #as >
{Propiedad de listas, elemento neutro de la suma}
genf.[ ].0.xs = < Max as, bs, cs : xs = as ++ bs ++ cs ∧ as = cs : #as >
{Definición de f.xs}
genf.[ ].0.xs = f.xs

Ya probamos que nuestra función generalizada es un caso particular de nuestra función original. Derivemos la misma y encontremos un programa.

genf.ks.z.[ ] = < Max as, bs, cs : [ ] = as ++ bs ++ cs ∧ ks ++ as = cs : z + #[ ] >
{Propiedad de listas}
…
{Rango unitario}
z + #[ ]
{Definición de cardinal de lista}
z

genf.ks.z.(x.xs) = < Max as, bs, cs : (x.xs) = as ++ bs ++ cs ∧ ks ++ as = cs : z + #as >
{Tercero excluido con as = [ ] v as ≠ [ ]}
< Max as, bs, cs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ bs ++ cs ∧ ks ++ as = cs : z + #as >
{Distributividad y partición de rango}
< Max as, bs, cs : as = [ ] ^ (x.xs) = as ++ bs ++ cs ∧ ks ++ as = cs : z + #as > max
< Max as, bs, cs : as ≠ [ ] ^ (x.xs) = as ++ bs ++ cs ∧ ks ++ as = cs : z + #as >
{Eliminación de variable en el primer término con as = [ ]}
< Max bs, cs : (x.xs) = [ ] ++ bs ++ cs ∧ ks ++ [ ] = cs : z + #[ ] > max
< Max as, bs, cs : as ≠ [ ] ^ (x.xs) = as ++ bs ++ cs ∧ ks ++ as = cs : z + #[ ] >
{Propiedad de listas}
< Max bs, cs : (x.xs) = bs ++ cs ∧ (ks = cs) : z + 0 > max
< Max as, bs, cs : as ≠ [ ] ^ (x.xs) = as ++ bs ++ cs ∧ ks ++ as = cs : z + #as >
{Neutro de la suma, eliminación de variable con cs}
< Max bs : (x.xs) = bs ++ ks : z > max
< Max as, bs, cs : as ≠ [ ] ^ (x.xs) = as ++ bs ++ cs ∧ ks ++ as = cs : z + #as >
{Término Constante}
z max < Max as, bs, cs : as ≠ [ ] ^ (x.xs) = as ++ bs ++ cs ∧ ks ++ as = cs : z + #as >
{Cambio de variable con as ←(a.as) válido por tercero excluido}
z max
< Max a,as, bs, cs : (a.as) ≠ [ ] ^ (x.xs) = (a.as) ++ bs ++ cs ∧ ks ++ (a.as) = cs : z + #(a.as) >
{Propiedad de Listas, cardinal de listas}

< Max a,as, bs, cs : True ^ (x = a) ^ (xs = as ++ bs ++ cs) ∧ (ks ++ (a.as) = cs) : z + 1 + #as >

{Neutro de la conjunción, eliminación de variable con x = a}

z max < Max as, bs, cs : (xs = as ++ bs ++ cs) ∧ (ks ++ (x.as) = cs) : (z + 1) + #as >

{Aplicamos la ayuda}

z max < Max as, bs, cs : (xs = as ++ bs ++ cs) ∧ (ks ++ [x] ++ as = cs) : (z + 1) + #as >

{Aplicamos de nuevo la ayuda}

z max < Max as, bs, cs : (xs = as ++ bs ++ cs) ∧ ([x].ks ++ as = cs) : (z + 1) + #as >

{Hipótesis inductiva}

z max genf.([x].ks).(z + 1).xs

Programa final, anotado:

f.[ ] = 0
f.(x.xs) = 0 max genf.[ ].0.xs
genf.ks.z.[ ] = z
genf.ks.z.(x.xs) = z max genf.([x].ks).(z + 1).xs

- g.n.xs = < N as, cs : xs = as ++ cs : prod.as ≤ n >

Derivemos g mediante inducción en xs.

g.n.[ ] = < N as, cs : [ ] = as ++ cs : prod.[ ] ≤ n >

{Propiedad de listas y definición de prod. [ ]}

g.n.[ ] = < N as, cs : as = [ ] ^ cs = [ ] : 1 ≤ n >

{Anidado}

g.n.[ ] = < N as: as = [ ] < N cs : cs = [ ] : 1 ≤ n > >

{Rango unitario}

< N cs : cs = [ ] : 1 ≤ n >

{Rango unitario}

(1 ≤ n → 1
¬ (1 ≤ n) → 0
)

g.n.(x.xs) = < N as, cs : (x.xs) = as ++ cs : prod.as ≤ n >

{Tercero excluido}

g.n.(x.xs) = < N as, cs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ cs : prod.as ≤ n >

{Distributividad}

g.n.(x.xs) = < N as, cs : (as = [ ] ^ (x.xs) = as ++ cs) v (as ≠ [ ] ^ (x.xs) = as ++ cs) : prod.as ≤ n >

{Partición de rango}

g.n.(x.xs) = < N as, cs : as = [ ] ^ (x.xs) = as ++ cs : prod.as ≤ n > +
< N as, cs : as ≠ [ ] ^ (x.xs) = as ++ cs : prod.as ≤ n >

{Eliminación de variable con as = [ ], as ← (a.as) válido por tercero excluido}

$$g.n.(x.xs) = \langle N\ cs : (x.xs) = [\,] \mathbin{++} cs : prod.[\,] \leq n \rangle +$$
$$\langle N\ a, as, cs : (a.as) \neq [\,] \wedge (x.xs) = (a.as) \mathbin{++} cs : prod.(a.as) \leq n \rangle$$
{Propiedad de listas, definición de prod.[ ] y de prod.(a.as)
$$g.n.(x.xs) = \langle N\ cs : (x.xs) = cs : 1 \leq n \rangle +$$
$$\langle N\ a, as, cs : True \wedge (x = a) \wedge xs = as \mathbin{++} cs : a * prod.as \leq n \rangle$$
{Neutro de la conjunción, eliminación de variable con x = a}
$$g.n.(x.xs) = \langle N\ cs : (x.xs) = cs : 1 \leq n \rangle +$$
$$\langle N\ as, cs : xs = as \mathbin{++} cs : x * prod.as \leq n \rangle$$
{Rango unitario}
$$\begin{array}{l}(1 \leq n \rightarrow 1 + \langle N\ as, cs : xs = as \mathbin{++} cs : x * prod.as \leq n \rangle \\ \ \neg (1 \leq n) \rightarrow \langle N\ as, cs : xs = as \mathbin{++} cs : x * prod.as \leq n \rangle \\ ) \end{array}$$
{No puedo aplicar hipótesis inductiva, por ende generalizo}

$$geng.n.k.xs = \langle N\ as, cs : xs = as \mathbin{++} cs : k * prod.as \leq n \rangle$$

$$geng.n.1.xs = \langle N\ as, cs : xs = as \mathbin{++} cs : 1 * prod.as \leq n \rangle$$
{Neutro del producto}
$$geng.n.1.xs = \langle N\ as, cs : xs = as \mathbin{++} cs : prod.as \leq n \rangle$$
{Definición de g.n.xs}
$$geng.n.1.xs = g.n.xs$$

Derivemos la función generalizada:

$$geng.n.k.[\,] = \langle N\ as, cs : [\,] = as \mathbin{++} cs : k * prod.as \leq n \rangle$$
{Propiedad de listas}
$$geng.n.k.[\,] = \langle N\ as, cs : as = [\,] \wedge cs = [\,] : k * prod.as \leq n \rangle$$
{Anidado}
$$geng.n.k.[\,] = \langle N\ as : as = [\,] : \langle N\ cs : cs = [\,] : k * prod.as \leq n \rangle\rangle$$
{Rango Unitario}
$$geng.n.k.[\,] = \langle N\ cs : cs = [\,] : k * prod.[\,] \leq n \rangle$$
{Rango unitario, y def de prod. [ ]}
$$\begin{array}{l}geng.n.k.[\,] = (k * 1 \leq n \rightarrow 1 \\ \qquad\qquad\qquad \neg (k * 1 \leq n) \rightarrow 0 \\ \qquad\qquad\qquad) \end{array}$$

Derivación de la función x.xs

$$geng.n.k.(x.xs) = \langle N\ as, cs : (x.xs) = as \mathbin{++} cs : k * prod.as \leq n \rangle$$
{Tercero excluido}
$$geng.n.k.(x.xs) = \langle N\ as, cs : (as = [\,] \vee as \neq [\,]) \wedge (x.xs) = as \mathbin{++} cs : k * prod.as \leq n \rangle$$
{Distributividad}
$$geng.n.k.(x.xs) = \langle N\ as, cs : as = [\,] \wedge (x.xs) = as \mathbin{++} cs : k * prod.as \leq n \rangle +$$
$$geng.n.k.(x.xs) = \langle N\ as, cs : as \neq [\,] \wedge (x.xs) = as \mathbin{++} cs : k * prod.as \leq n \rangle$$

{Eliminación de variable con as = [ ]}
geng.n.k.(x.xs) = < N cs : (x.xs) = [ ] ++ cs : k * prod.[ ] ≤ n > +
< N as, cs : as ≠ [ ] ^ (x.xs) = as ++ cs : k * prod.as ≤ n >
{Propiedad de listas y def de prod.[ ]}
geng.n.k.(x.xs) = < N cs : (x.xs) = cs : k * 1 ≤ n > +
< N as, cs : as ≠ [ ] ^ (x.xs) = as ++ cs : k * prod.as ≤ n >
{Cambio de variable en el segundo termino as ← a.as}
geng.n.k.(x.xs) = < N cs : (x.xs) = cs : k * 1 ≤ n > +
< N a, as, cs : (a.as) ≠ [ ] ^ (x.xs) = (a.as) ++ cs : k * prod.(a.as) ≤ n >
{Propiedad de listas y def de prod.(a.as)}
geng.n.k.(x.xs) = < N cs : (x.xs) = cs : k * 1 ≤ n > +
< N a, as, cs : True ^ (x = a) ^ xs = as ++ cs : k * a * prod.(as) ≤ n >
{Neutro de la Conjunción, eliminación con a = x}
geng.n.k.(x.xs) = < N cs : (x.xs) = cs : k * 1 ≤ n > +
< N as, cs : xs = as ++ cs : (k * x) * prod.(as) ≤ n >
{Rango unitario en el primer término, Hipótesis inductiva}
geng.n.k.(x.xs) = (k * 1 ≤ n → 1 + geng.n.(k * x).xs
¬ (k * 1 ≤ n) → geng.n.(k * x).xs
)

Programa final, anotado :
g.n. [ ] = 1 ≤ n → 1
¬ (1 ≤ n) → 0
g.n.(x.xs) = geng.n.1.xs
geng.n.k.[ ] = k * 1 ≤ n → 1
¬ (k * 1 ≤ n) → 0
geng.n.k.(x.xs) = k * 1 ≤ n → 1 + geng.n.(k * x).xs
¬ (k * 1 ≤ n) → geng.n.(k * x).xs

- h.xs = < Max as, bs : xs = as + +bs ∧ pares.as : sum.as >

h.[ ] = < Max as, bs : [ ] = as + +bs ∧ pares.as : sum.as >
{Propiedad de listas}
h.[ ] = < Max as, bs : as = [ ] ^ bs = [ ] ∧ pares.as : sum.as >
{Eliminación de variable con as = [ ]}
h.[ ] = < Max bs : bs = [ ] ∧ pares.[ ] : sum.[ ] >
{Definición de pares.[ ] y de sum.[ ]}
h.[ ] = < Max bs : bs = [ ] ∧ True : 0 >
{Neutro de la conjunción}
h.[ ] = < Max bs : bs = [ ] : 0 >
{Rango unitario}
h.[ ] = < Max bs : bs = [ ] : 0 >
h.[ ] = 0

h.(x.xs) = < Max as, bs : (x.xs) = as ++ bs ∧ pares.as : sum.as >
{Tercero excluido con as = [ ] v as ≠ [ ]}

h.(x.xs) = < Max as, bs : (as = [ ] v as ≠ [ ]) ^ (x.xs) = as + +bs ∧ pares.as : sum.as >

{Distributividad y partición de rango}

h.(x.xs) = < Max as, bs : as = [ ] ^ (x.xs) = as ++ bs ∧ pares.as : sum.as > max
< Max as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs ∧ pares.as : sum.as >

{Eliminación de variable as en el primer término}

h.(x.xs) = < Max bs : (x.xs) = [ ] ++ bs ∧ pares.[ ] : sum.[ ] > max
< Max as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs ∧ pares.as : sum.as >

{Def de sum [ ] y pares [ ], propiedades de listas}

h.(x.xs) = < Max bs : (x.xs) = bs ∧ True : 0 > max
< Max as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs ∧ pares.as : sum.as >

{Neutro de la conjunción}

h.(x.xs) = < Max bs : (x.xs) = bs : 0 > max
< Max as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs ∧ pares.as : sum.as >

{Rango Unitario}

h.(x.xs) = 0 max < Max as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs ∧ pares.as : sum.as >

{Cambio de variable as ← a.as, válido por tercero excluido}

h.(x.xs) = 0 max < Max a, as, bs : (a.as) ≠ [ ] ^ (x.xs) = (a.as) ++ bs ∧ pares.(a.as) : sum.(a.as) >

Aritmética, definición de pares.(a.as) y sum.(a.as)}

h.(x.xs) = 0 max < Max a, as, bs : True ^ (x.xs) = (a.as) ++ bs ∧ par.a ∧ pares.as : a + sum.as >

{Propiedad de listas, elemento neutro de la conjunción}

h.(x.xs) = 0 max < Max a, as, bs : (x = a) ^ xs = as ++ bs ∧ par.a ∧ pares.as : a + sum.as >

{Eliminación de variable con x = a}

h.(x.xs) = 0 max < Max as, bs : xs = as ++ bs ∧ par.x ∧ pares.as : x + sum.as >

{Propongo un análisis por casos basado en par.x}

Caso par.x

h.(x.xs) = 0 max < Max as, bs : xs = as ++ bs ∧ True ∧ pares.as : x + sum.as >

{elemento neutro de la conjunción}

h.(x.xs) = 0 max < Max as, bs : xs = as ++ bs ∧ pares.as : x + sum.as >

{Aquí debo generalizar para aplicar hipótesis inductiva}

Caso ¬(par.x)

h.(x.xs) = 0 max < Max as, bs : xs = as ++ bs ∧ False ∧ pares.as : x + sum.as >

{Elemento absorbente de la conjunción}x

h.(x.xs) = 0 max < Max as, bs : False : x + sum.as >

{Rango vacío}

h.(x.xs) = 0 max -inf

{Aritmética}

0

Introducimos $hgen.k.xs = \langle \text{Max } as, bs : xs = as \mathbin{++} bs \wedge pares.as : k + sum.as \rangle$ y probemos que $h.xs$ es un caso particular de de $hgen.k.xs$

$$hgen.0.xs = \langle \text{Max } as, bs : xs = as \mathbin{++} bs \wedge pares.as : 0 + sum.as \rangle$$

{Neutro de la suma}

$$hgen.0.xs = \langle \text{Max } as, bs : xs = as \mathbin{++} bs \wedge pares.as : sum.as \rangle$$

{Definición de h.xs}

$$hgen.0.xs = h.xs$$

$$hgen.k.[\ ] = \langle \text{Max } as, bs : [\ ] = as \mathbin{++} bs \wedge pares.as : k + sum.as \rangle$$

{Propiedad de listas}

$$hgen.k.[\ ] = \langle \text{Max } as, bs : as = [\ ] \wedge bs = [\ ] \wedge pares.as : k + sum.as \rangle$$

{Eliminación de as = [ ]}

$$hgen.k.[\ ] = \langle \text{Max } bs : bs = [\ ] \wedge pares.[\ ] : k + sum.[\ ] \rangle$$

{Definición de pares.[ ] y de sum. [ ]}

$$hgen.k.[\ ] = \langle \text{Max } bs : bs = [\ ] \wedge True : k + 0 \rangle$$

{Neutro de la suma, y elemento neutro de la conjunción}

$$hgen.k.[\ ] = \langle \text{Max } bs : bs = [\ ] : k \rangle$$

{Rango unitario}

$$hgen.k.[\ ] = k$$

$$hgen.k.(x.xs) = \langle \text{Max } as, bs : (x.xs) = as \mathbin{++} bs \wedge pares.as : x + sum.as \rangle$$

{Tercero excluido}

$$hgen.k.(x.xs) = \langle \text{Max } as, bs : (as = [\ ] \vee as \neq [\ ]) \wedge (x.xs) = as \mathbin{++} bs \wedge pares.as : x + sum.as \rangle$$

{Distributividad y partición de rango}

$$hgen.k.(x.xs) = \langle \text{Max } as, bs : as \neq [\ ] \wedge (x.xs) = as \mathbin{++} bs \wedge pares.as : k + sum.as \rangle$$
$$\text{max } \langle \text{Max } as, bs : as = [\ ] \wedge (x.xs) = as \mathbin{++} bs \wedge pares.as : k + sum.as \rangle$$

{Eliminación de variable en el segundo término, reemplazo de as ←(a.as)}

$$hgen.k.(x.xs) = \langle \text{Max } a, as, bs : (a.as) \neq [\ ] \wedge (x.xs) = (a.as) \mathbin{++} bs \wedge pares.(a.as) : k + sum.(a.as) \rangle \text{ max } \langle \text{Max } bs : (x.xs) = [\ ] \mathbin{++} bs \wedge pares.[\ ] : k + sum.[\ ] \rangle$$

{Propiedad de listas en ambos términos, definición de pares.[ ] y de sum.[ ]}

$$hgen.k.(x.xs) = \langle \text{Max } a, as, bs : True \wedge (x.xs) = a.(as \mathbin{++} bs) \wedge pares.(a.as) : k + sum.(a.as) \rangle \text{ max } \langle \text{Max } bs : (x.xs) = bs \wedge True : k + 0 \rangle$$

{Elemento neutro de la conjunción, propiedad de listas}

$$hgen.k.(x.xs) = \langle \text{Max } a, as, bs : (x = a) \wedge xs = as \mathbin{++} bs \wedge pares.(a.as) : k + sum.(a.as) \rangle \text{ max } \langle \text{Max } bs : (x.xs) = bs : k + 0 \rangle$$

{Eliminación de variable con a = x, rango unitario}

$$hgen.k.(x.xs) = \langle \text{Max } as, bs : xs = as \mathbin{++} bs \wedge pares.(x.as) : k + sum.(x.as) \rangle \text{ max } k$$

{Definición de pares.(x.as) y de sum.(x.as)}

$$hgen.k.(x.xs) = \langle \text{Max } as, bs : xs = as \mathbin{++} bs \wedge par.x \wedge pares.as : (k + x) + sum.as \rangle \text{ max } k$$

{Análisis por casos}

Caso par.x

$$\langle \text{Max } as, bs : xs = as \mathbin{++} bs \wedge True \wedge pares.as : (k + x) + sum.as \rangle \max k$$
{Elemento neutro de la conjunción}
$$\langle \text{Max } as, bs : xs = as \mathbin{++} bs \wedge pares.as : (k + x) + sum.as \rangle \max k$$
{Hipótesis inductiva}
$$hgen.(k + x).xs \max k$$

Caso ¬(par.x)

$$\langle \text{Max } as, bs : xs = as \mathbin{++} bs \wedge False \wedge pares.as : (k + x) + sum.as \rangle \max k$$
{Elemento absorbente de la conjunción}
$$\langle \text{Max } as, bs : False : (k + x) + sum.as \rangle \max k$$
{Rango vacío}
$$-inf \max k$$
{Aritmética}
$$k$$

Programa final anotado :
h.[ ] = 0
h.(x.xs) = hgen.0.xs
hgen.k.[ ] = k
hgen.k.(x.xs) = (par.x →hgen.(k + x).xs max k
¬(par.x) → k
)

- $ham.xs.ys = \langle N\ i : 0 \leq i < \#xs \min \#ys : xs!i \neq ys!i \rangle$

Propongo derivar mediante inducción en ambas listas, por ende tendremos varios casos bases.

$$ham.[\ ].ys = \langle N\ i : 0 \leq i < \#[\ ] \min \#ys : [\ ]!i \neq ys!i \rangle$$
{Cardinal de lista vacía}
$$ham.[\ ].ys = \langle N\ i : 0 \leq i < 0 \min \#ys : [\ ]!i \neq ys!i \rangle$$
{Aritmética en el rango, 0 es el menor cardinal posible en una lista}
$$ham.[\ ].ys = \langle N\ i : 0 \leq i < 0 : [\ ]!i \neq ys!i \rangle$$
{Aritmética}
$$ham.[\ ].ys = \langle N\ i : False : [\ ]!i \neq ys!i \rangle$$
{Rango vacío}
$$ham.[\ ].ys = 0$$

De aquí deducimos que los casos ham.[ ].ys = ham.xs.[ ] = ham.[ ].[ ] son iguales y devuelven 0. Derivemos entonces el caso inductivo:
$$ham.(x.xs).(y.ys) = \langle N\ i : 0 \leq i < \#(x.xs) \min \#(x.xs) : (x.xs)!i \neq (y.ys)!i \rangle$$
{Definición de cardinal de lista}

$$ham.(x.xs).(y.ys) = < N\ i : 0 \leq i < (1 + \#xs)\ min\ (1 + \#ys) : (x.xs)!i \neq (y.ys)!i >$$
$$\{\text{Aritmética en el rango}\}$$
$$ham.(x.xs).(y.ys) = < N\ i : i = 0 \lor 1 \leq i < (1 + \#xs)\ min\ (1 + \#ys) : (x.xs)!i \neq (y.ys)!i >$$
$$\{\text{Partición de rango}\}$$
$$ham.(x.xs).(y.ys) = < N\ i : 1 \leq i < (1 + \#xs)\ min\ (1 + \#ys) : (x.xs)!i \neq (y.ys)!i > +$$
$$< N\ i : i = 0 : (x.xs)!i \neq (y.ys)!i >$$
$$\{\text{Cambio de variable } i \leftarrow i + 1\}$$
$$ham.(x.xs).(y.ys) = < N\ i : 1 \leq i + 1 < (1 + \#xs)\ min\ (1 + \#ys) : (x.xs)!i + 1 \neq (y.ys)!i + 1 >$$
$$+ < N\ i : i = 0 : (x.xs)!i \neq (y.ys)!i >$$
$$\{\text{Aritmética en el rango, definición de indexación}\}$$
$$ham.(x.xs).(y.ys) = < N\ i : 0 \leq i < \#xs\ min\ (1 + \#ys) : xs!i \neq ys!i > +$$
$$< N\ i : i = 0 : (x.xs)!i \neq (y.ys)!i >$$
$$\{\text{No puedo aplicar hipótesis inductiva, es necesario generalizar}\}$$

Propongo $genham.xs.ys.k = < N\ i : 0 \leq i < \#xs\ min\ k + \#ys : xs!i \neq ys!i >$ y pruebo que ham.xs.ys es un caso particular de mi nueva función generalizada.

$$genham.xs.ys.k = < N\ i : 0 \leq i < \#xs\ min\ k + \#ys : xs!i \neq ys!i >$$
$$\{\text{Elijo } k \leftarrow 0\}$$
$$genham.xs.ys.0 = < N\ i : 0 \leq i < \#xs\ min\ 0 + \#ys : xs!i \neq ys!i >$$
$$\{\text{Elemento neutro de la suma}\}$$
$$genham.xs.ys.0 = < N\ i : 0 \leq i < \#xs\ min\ \#ys : xs!i \neq ys!i >$$
$$\{\text{Definición de ham.xs.ys}\}$$
$$genham.xs.ys.0 = ham.xs.ys$$

Ahora derivó genham.xs.ys.k :
Tener en cuenta que los casos bases son análogos y triviales a los anteriores, no los vamos a ver.

$$genham.(x.xs).(y.ys).k = < N\ i : 0 \leq i < \#(x.xs)\ min\ k + \#(y.ys) : (x.xs)!i \neq (y.ys)!i >$$
$$\{\text{Definición de cardinal de listas}\}$$
$$genham.(x.xs).(y.ys).k = < N\ i : 0 \leq i < 1 + \#xs\ min\ k + 1 + \#ys : (x.xs)!i \neq (y.ys)!i >$$
$$\{\text{Aritmética en el rango, asociatividad de la suma}\}$$
$$genham.(x.xs).(y.ys).k = < N\ i : i = 0 \lor 1 \leq i < 1 + \#xs\ min\ (k + 1) + \#ys : (x.xs)!i \neq$$
$$(y.ys)!i >$$
$$\{\text{Partición de rango}\}$$
$$genham.(x.xs).(y.ys).k = < N\ i : 1 \leq i < 1 + \#xs\ min\ (k + 1) + \#ys : (x.xs)!i \neq (y.ys)!i > +$$
$$< N\ i : i = 0 : (x.xs)!i \neq (y.ys)!i >$$
$$\{\text{Cambio de variable } i \leftarrow i + 1\}$$
$$genham.(x.xs).(y.ys).k = < N\ i : 1 \leq i + 1 < 1 + \#xs\ min\ (k + 1) + \#ys : (x.xs)!i + 1 \neq$$
$$(y.ys)!i + 1 > +$$
$$< N\ i : i = 0 : (x.xs)!i \neq (y.ys)!i >$$
$$\{\text{Aritmética en el rango, definición de indexación}\}$$
$$genham.(x.xs).(y.ys).k = < N\ i : 0 \leq i < \#xs\ min\ (k + 1) + \#ys : xs!i \neq ys!i > +$$

< N i : i = 0 : (x.xs)!i ≠ (y.ys)!i >
{Hipótesis inductiva}
genham.(x.xs).(y.ys).k = genham.xs.ys.(k + 1) + < N i : i = 0 : (x.xs)!i ≠ (y.ys)!i >
{Rango unitario}
((x ≠ y) → genham.xs.ys.(k + 1) + 1
¬ (x ≠ y) → genham.xs.ys.(k + 1)
)

Programa final, anotado :
ham.[ ].ys = 0
ham.xs.[ ] = 0
ham.[ ].[ ] = 0
ham.(x.xs).(y.ys) = genham.xs.ys.0
genham.[ ].ys.k = 0
genham.xs.[ ].k = 0
genham.(x.xs).(y.ys).k = (x ≠ y) → genham.xs.ys.(k + 1) + 1
¬ (x ≠ y) → genham.xs.ys.(k + 1)

- pj.xs = < ∃ as, bs : xs = as ++ bs : sum.as = sum.bs >

Derivamos la función mediante inducción en xs. Con el caso base xs = [ ] y el caso inductivo xs = (x.xs)

pj.[ ] = < ∃ as, bs : [ ] = as ++ bs : sum.as = sum.bs >
{Propiedad de listas}
pj.[ ] = < ∃ as, bs : as = [ ] ^ bs = [ ] : sum.as = sum.bs >
{Anidado}
pj.[ ] = < ∃ as : as = [ ] : < ∃ bs : bs = [ ] : sum.as = sum.bs > >
{Rango unitario}
< ∃ bs : bs = [ ] : sum.[ ] = sum.bs >
{Rango unitario}
sum.[ ] = sum.[ ]
{Definición de sum.[ ]}
0 = 0
{Aritmética}
True

pj.(x.xs) = < ∃ as, bs : (x.xs) = as ++ bs : sum.as = sum.bs >
{Tercero excluido}

$$pj.(x.xs) = < \exists\ as, bs : (as = [\,] \lor as \neq [\,]) \land (x.xs) = as \mathbin{++} bs : sum.as = sum.bs >$$

{Distributividad y partición de rango}

$$pj.(x.xs) = < \exists\ as, bs : as = [\,] \land (x.xs) = as \mathbin{++} bs : sum.as = sum.bs > \lor$$
$$< \exists\ as, bs : as \neq [\,] \land (x.xs) = as \mathbin{++} bs : sum.as = sum.bs >$$

{Eliminación de variable con as = [ ], reemplazo de as ←(a.as)}

$$pj.(x.xs) = < \exists\ bs : (x.xs) = [\,] \mathbin{++} bs : sum.[\,] = sum.bs > \lor$$
$$< \exists\ a, as, bs : (a.as) \neq [\,] \land (x.xs) = (a.as) \mathbin{++} bs : sum.(a.as) = sum.bs >$$

{Propiedad de listas, definición de sum.[ ]}

$$pj.(x.xs) = < \exists\ bs : (x.xs) = bs : 0 = sum.bs > \lor$$
$$< \exists\ a, as, bs : True \land (x.xs) = (a.as) \mathbin{++} bs : sum.(a.as) = sum.bs >$$

{Elemento neutro de la conjunción, propiedad de listas}

$$pj.(x.xs) = < \exists\ bs : (x.xs) = bs : 0 = sum.bs > \lor$$
$$< \exists\ a, as, bs : (x = a) \land xs = as \mathbin{++} bs : sum.(a.as) = sum.bs >$$

{Rango unitario, eliminación de variable con a = x}

$$pj.(x.xs) = 0 = sum.(x.xs) \lor$$
$$< \exists\ as, bs : xs = as \mathbin{++} bs : sum.(x.as) = sum.bs >$$

{Definición de sum.(x.as)}

$$pj.(x.xs) = (0 = sum.(x.xs)) \lor < \exists\ as, bs : xs = as \mathbin{++} bs : x + sum.as = sum.bs >$$

{Es necesario generalizar ya que no puedo aplicar hipótesis inductiva}

$$genpj.k.xs = < \exists\ as, bs : xs = as \mathbin{++} bs : k + sum.as = sum.bs >$$

Derivemos la función generalizada pero antes probemos que pj.xs es un caso particular de nuestra función generalizada.

$$genpj.0.xs$$

{Especificación}

$$genpj.0.xs = < \exists\ as, bs : xs = as \mathbin{++} bs : 0 + sum.as = sum.bs >$$

{Neutro de la suma}

$$genpj.0.xs = < \exists\ as, bs : xs = as \mathbin{++} bs : sum.as = sum.bs >$$

{Definición de pj.xs}

$$genpj.0.xs = pj.xs$$

$$genpj.k.[\,] = < \exists\ as, bs : [\,] = as \mathbin{++} bs : k + sum.as = sum.bs >$$

{Propiedad de listas}

$$genpj.k.[\,] = < \exists\ as, bs : as = [\,] \land bs = [\,] : k + sum.as = sum.bs >$$

{Anidado}

$$genpj.k.[\,] = < \exists\ as : as = [\,] : < \exists\ bs : bs = [\,] : k + sum.as = sum.bs > >$$

{Rango unitario}

$$genpj.k.[\,] = < \exists\ bs : bs = [\,] : k + sum.[\,] = sum.bs >$$

{Rango unitario}

$$k + sum.[\,] = sum.[\,]$$

{Definición de sum.[ ]}

$$k + 0 = 0$$

{Aritmética}

$(k = 0)$

$genpj.k.(x.xs) = \langle \exists\ as, bs : (x.xs) = as \mathbin{++} bs : k + sum.as = sum.bs$

{Tercero excluido}

$genpj.k.(x.xs) = \langle \exists\ as, bs : (as = [\,] \lor as \neq [\,]) \land (x.xs) = as \mathbin{++} bs : k + sum.as = sum.bs \rangle$

{Distributividad y partición de rango}

$genpj.k.(x.xs) = \langle \exists\ as, bs : as = [\,] \land (x.xs) = as \mathbin{++} bs : k + sum.as = sum.bs \rangle \lor$
$\langle \exists\ as, bs : as \neq [\,] \land (x.xs) = as \mathbin{++} bs : k + sum.as = sum.bs \rangle$

{Eliminacion de variable con as = [ ], reemplazo de as ←a.as}

$genpj.k.(x.xs) = \langle \exists\ bs : (x.xs) = [\,] \mathbin{++} bs : k + sum.[\,] = sum.bs \rangle \lor$
$\langle \exists\ a, as, bs : (a.as) \neq [\,] \land (x.xs) = as \mathbin{++} bs : k + sum.(a.as) = sum.bs \rangle$

{Propiedad de listas, definición de sum.[ ] y sum.(a.as)}

$genpj.k.(x.xs) = \langle \exists\ bs : (x.xs) = bs : k + 0 = sum.bs \rangle \lor$
$\langle \exists\ a, as, bs : True \land (x.xs) = as \mathbin{++} bs : k + a + sum.as = sum.bs \rangle$

{Rango unitario, aritmética, propiedad de listas y neutro de la conjunción}

$genpj.k.(x.xs) = (k = sum.bs) \lor \langle \exists\ a, as, bs : (x.xs) = as \mathbin{++} bs : (k + a) + sum.as = sum.bs \rangle$

{Propiedad de listas y eliminación de variable con a = x}

$genpj.k.(x.xs) = (k = sum.(x.xs)) \lor \langle \exists\ as, bs : xs = as \mathbin{++} bs : (k + x) + sum.as = sum.bs \rangle$

{Hipótesis inductiva y definición de sum.(x.xs)}

$genpj.k.(x.xs) = (k = x + sum.xs) \lor genpj.(k + x).(xs)$

Programa final, anotado :
$pj.[\,] = True$
$pj.(x.xs) = genpj.0.xs$
$genpj.k.[\,] = (k = 0)$
$genpj.k.(x.xs) = (k = x + sum.xs) \lor genpj.(k + x).(xs)$

- $f.xs = \langle \exists\ i : 0 \leq i < \#xs : xs!i = 2 * i \rangle$

Derivamos mediante inducción en xs con caso base xs = [ ] y caso inductivo (x.xs)

$f.[\,] = \langle \exists\ i : 0 \leq i < \#[\,] : [\,]!i = 2 * i \rangle$

{Definición de cardinal de lista vacía}

$f.[\,] = \langle \exists\ i : 0 \leq i < 0 : [\,]!i = 2 * i \rangle$

{Aritmética}

$f.[\,] = \langle \exists\ i : False : [\,]!i = 2 * i \rangle$

{Rango vacío}

False

$f.(x.xs) = \langle \exists\ i : 0 \leq i < \#(x.xs) : (x.xs)!i = 2 * i \rangle$

{Definición de cardinal de lista}

$$f.(x.xs) = < \exists\ i : 0 \leq i < 1 + \#xs : (x.xs)!i = 2 * i >$$
{Aritmética en el rango}
$$f.(x.xs) = < \exists\ i : i = 0 \vee 1 \leq i < 1 + \#xs : (x.xs)!i = 2 * i >$$
{Partición de rango}
$$f.(x.xs) = < \exists\ i : 1 \leq i < 1 + \#xs : (x.xs)!i = 2 * i > \vee$$
$$< \exists\ i : i = 0 : (x.xs)!i = 2 * i >$$
{Rango unitario y cambio de variable $i \leftarrow i + 1$}
$$f.(x.xs) = < \exists\ i : 1 \leq i + 1 < 1 + \#xs : (x.xs)!i + 1 = 2 * (i + 1) > \vee (x.xs)!0 = 2 * 0$$
{Aritmética en el rango, definición de indexación}
$$f.(x.xs) = < \exists\ i : 0 \leq i < \ + \#xs : xs!i \ = 2 * (i + 1) > \vee (x = 0)$$
{No puedo aplicar hipótesis inductiva, debo generalizar}

$$genf.k.xs = < \exists\ i : 0 \leq i < \ + \#xs : xs!i \ = 2 * i + k >$$

Probemos que f.xs es un caso particular de genf.k.xs

$$genf.0.xs = < \exists\ i : 0 \leq i < \ + \#xs : xs!i \ = 2 * i + 0 >$$
{Elemento neutro de la suma}
$$genf.0.xs = < \exists\ i : 0 \leq i < \ + \#xs : xs!i \ = 2 * i >$$
{Especificación de f.xs}
$$genf.0.xs = f.xs$$

$$genf.k.[\ ] = < \exists\ i : 0 \leq i < \ + \#[\ ] : [\ ]!i \ = 2 * i + k >$$
{Definición de cardinal vacío}
$$genf.k.[\ ] = < \exists\ i : 0 \leq i < 0 : [\ ]!i \ = 2 * i + k >$$
{Aritmética en el rango}
$$genf.k.[\ ] = < \exists\ i : False : [\ ]!i \ = 2 * i + k >$$
{Rango vacío}
$$genf.k.[\ ] = False$$

$$genf.k.(x.xs) = < \exists\ i : 0 \leq i < \#(x.xs) : (x.xs)!i \ = 2 * i + k >$$
{Definición de Cardinal}
$$genf.k.(x.xs) = < \exists\ i : 0 \leq i < 1 + \#xs : (x.xs)!i \ = 2 * i + k >$$
{Aritmética en el Rango}
$$genf.k.(x.xs) = < \exists\ i : i = 0 \ \wedge 1 \leq i < 1 + \#xs : (x.xs)!i \ = 2 * i + k >$$
{Partición de Rango}
$$genf.k.(x.xs) = < \exists\ i : i = 0 : (x.xs)!i \ = 2 * i + k > \vee$$
$$< \exists\ i : 1 \leq i < 1 + \#xs : (x.xs)!i \ = 2 * i + k >$$
{Rango unitario, cambio de variable $i \leftarrow i + 1$}
$$genf.k.(x.xs) = ((x.xs)!0 = 2 * 0 + k) \ \vee$$
$$< \exists\ i : 1 \leq i +1 < 1 + \#xs : (x.xs)!i+1 \ = 2 * i+1 + k >$$
{Indexación, aritmetica, aritmetica en el rango}
$$genf.k.(x.xs) = (x = k) \ \vee < \exists\ i : 0 \leq i < \#xs : xs.i = 2 * i + 1 + k >$$

{Hipótesis Inductiva}
genf.k.(x.xs) = (x = k) v genf.(1 + k).xs

Programa final, anotado :
f.[ ] = False
f.(x.xs) = genf.0.xs
genf.k.[ ] = False
genf.k(x.xs) = (x = k) v genf.(1 + k).xs

- f.xs = < ∃ as, bs : xs = as ++ bs : prod.as = sum.bs >

Derivemos f mediante inducción en xs con caso base xs = [ ] y caso inductivo xs = (x.xs).

f.[ ] = < ∃ as, bs : [ ] = as ++ bs : prod.as = sum.bs >
{Propiedad de listas}
f.[ ] = < ∃ as, bs : as = [ ] ^ bs = [ ] : prod.as = sum.bs >
{Anidado}
f.[ ] = < ∃ as : as = [ ] : < ∃ bs : bs = [ ] : prod.as = sum.bs > >
{Rango unitario}
< ∃ bs : bs = [ ] : prod.[ ] = sum.bs >
{Definición de prod.[ ]}
< ∃ bs : bs = [ ] : 1 = sum.bs >
{Rango unitario}
1 = sum.[ ]
{Definición de sum y aritmética}
False

f.(x.xs) = < ∃ as, bs : (x.xs) = as ++ bs : prod.as = sum.bs >
{Tercero excluido con (as = [ ] v as ≠ [ ])}
f.(x.xs) = < ∃ as, bs : (x.xs) = (as = [ ] v as ≠ [ ]) ^ (x.xs) = as ++ bs : prod.as = sum.bs >
{Distributividad}
f.(x.xs) = < ∃ as, bs : (as ≠ [ ] ^ (x.xs) = as ++ bs) v (as = [ ] ^ (x.xs) = as ++ bs : prod.as = sum.bs >
{Partición de rango}
f.(x.xs) = < ∃ as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs: prod.as = sum.bs > v
< ∃ as, bs : as = [ ] ^ (x.xs) = as ++ bs: prod.as = sum.bs >
{Eliminación de variable con as = [ ], reemplazo de as ←(a.as)
f.(x.xs) = < ∃ a, as, bs : (a.as) ≠ [ ] ^ (x.xs) = (a.as) ++ bs : prod.(a.as) = sum.bs > v
< ∃ bs : (x.xs) = [ ] ++ bs : prod.[ ] = sum.bs >
{Propiedad de listas, definición de prod.(a.as)}
f.(x.xs) = < ∃ a, as, bs : True ^ (x.xs) = a.(as ++ bs) : a * prod.as = sum.bs > v

$\langle \exists\ bs : (x.xs) = bs : prod.[\ ] = sum.bs \rangle$

{Elemento neutro de la conjunción, propiedad de listas y rango unitario}

$f.(x.xs) = \langle \exists\ a, as, bs : (x = a) \wedge xs = as +\!+\ bs : a * prod.as = sum.bs \rangle \vee$
$prod.[\ ] = sum.(x.xs)$

{Eliminación de variable con x = a, definición de prod.[ ] y de sum.(x.xs)}

$f.(x.xs) = \langle \exists\ as, bs : xs = as +\!+\ bs : x * prod.as = sum.bs \rangle \vee (1 = x + sum.xs)$

{No puedo aplicar hipótesis inductiva, por ende generalizo}

Introducción de la nueva función generalizada $genf.(x.xs) = \langle \exists\ as, bs : xs = as +\!+\ bs : x * prod.as = sum.bs \rangle$, luego vemos que f.xs es un caso particular de genf :

$genf.k.xs = \langle \exists\ as, bs : xs = as +\!+\ bs : k * prod.as = sum.bs \rangle$

{Elijo $k \leftarrow 1$}

$genf.1.xs = \langle \exists\ as, bs : xs = as +\!+\ bs : 1 * prod.as = sum.bs \rangle$

{Elemento neutro del producto}

$genf.1.xs = \langle \exists\ as, bs : xs = as +\!+\ bs : prod.as = sum.bs \rangle$

{Especificación de f.xs}

$genf.1.xs = f.xs$

Ahora, derivemos la función generalizada mediante inducción en xs con el caso base xs = [ ] y el caso inductivo xs = (x.xs) :

$genf.k.[\ ] = \langle \exists\ as, bs : [\ ] = as +\!+\ bs : k * prod.as = sum.bs \rangle$

{Propiedad de listas}

$genf.k.[\ ] = \langle \exists\ as, bs : as = [\ ] \wedge bs = [\ ] : k * prod.as = sum.bs \rangle$

{Anidado}

$genf.k.[\ ] = \langle \exists\ as : as = [\ ] : \langle \exists\ bs : bs = [\ ] : k * prod.as = sum.bs \rangle \rangle$

{Rango unitario}

$\langle \exists\ bs : bs = [\ ] : k * prod.[\ ] = sum.bs \rangle$

{Definición de prod.[ ]}

$\langle \exists\ bs : bs = [\ ] : k * 1 = sum.bs \rangle$

{Rango unitario y elemento neutro del producto}

$k = sum.[\ ]$

{Definición de sum.[ ]}

$k = 0$

$genf.k.(x.xs) = \langle \exists\ as, bs : (x.xs) = as +\!+\ bs : k * prod.as = sum.bs \rangle$

{Tercero excluido con (as = [ ] v as ≠ [ ])}

$genf.k.(x.xs) = \langle \exists\ as, bs : (x.xs) = (as = [\ ] \vee as \neq [\ ]) \wedge (x.xs) = as +\!+\ bs : k * prod.as$
$= sum.bs \rangle$

{Distributividad}

$genf.k.(x.xs) = \langle \exists\ as, bs : (as \neq [\ ] \wedge (x.xs) = as +\!+\ bs) \vee (as = [\ ] \wedge (x.xs) = as +\!+\ bs\ :$
$k * prod.as = sum.bs \rangle$

{Partición de rango}

$$genf.k.(x.xs) = \langle \exists\ as, bs : as \neq [\ ] \wedge (x.xs) = as +\!\!+\ bs : k * prod.as = sum.bs \rangle \vee$$
$$\langle \exists\ as, bs : as = [\ ] \wedge (x.xs) = as +\!\!+\ bs : k * prod.as = sum.bs \rangle$$
{Eliminación de variable con as = [ ] y reemplazo de as ←(a.as)}
$$genf.k.(x.xs) = \langle \exists\ a, as, bs : (a.as) \neq [\ ] \wedge (x.xs) = (a.as) +\!\!+\ bs : k * prod.(a.as) = sum.bs \rangle \vee$$
$$\langle \exists\ bs : (x.xs) = [\ ] +\!\!+\ bs : k * prod.[\ ] = sum.bs \rangle$$
{Propiedad de listas, definición de prod.(a.as) y prod.[ ]}
$$genf.k.(x.xs) = \langle \exists\ a, as, bs : True \wedge (x.xs) = a.(as +\!\!+\ bs) : (k * a) * prod.as = sum.bs \rangle \vee$$
$$\langle \exists\ bs : (x.xs) = bs : k * 1 = sum.bs \rangle$$
{Elemento neutro de la conjunción, propiedad de listas, rango unitario y aritmética}
$$genf.k.(x.xs) = \langle \exists\ a, as, bs : (x = a) \wedge xs = as +\!\!+\ bs : (k * a) * prod.as = sum.bs \rangle \vee$$
$$k = sum.(x.xs)$$
{Eliminación de variable con x = a, definición de sum.(x.xs)}
$$genf.k.(x.xs) = \langle \exists\ as, bs : xs = as +\!\!+\ bs : (k * x) * prod.as = sum.bs \rangle \vee$$
$$k = x + sum.xs$$
{Hipótesis inductiva}
$$genf.k.(x.xs) = genf.(k * x).xs \vee (k = x + sum.xs)$$

Programa final, anotado :

f.[ ] = False
f.(x.xs) = genf.1.xs
genf.k.[ ] = (k = 0)
genf.k.(x.xs) = genf.(k * x).xs v (k = x + sum.xs)

- $f.xs = \langle \exists\ as, bs : xs = as +\!\!+\ bs : \langle \Sigma\ i : 0 \leq i < \#as : as.i \rangle = \#bs \rangle$

Derivemos mediante inducción en el único parámetro de f, con el caso base xs = [ ] y el caso inductivo xs = (x.xs).

$$f.[\ ] = \langle \exists\ as, bs : [\ ] = as +\!\!+\ bs : \langle \Sigma\ i : 0 \leq i < \#as : as.i \rangle = \#bs \rangle$$
{Propiedad de listas}
$$f.[\ ] = \langle \exists\ as, bs : as = [\ ] \wedge bs = [\ ] : \langle \Sigma\ i : 0 \leq i < \#as : as.i \rangle = \#bs \rangle$$
{Anidado}
$$f.[\ ] = \langle \exists\ as : as = [\ ] : \langle \exists\ bs : bs = [\ ] : \langle \Sigma\ i : 0 \leq i < \#as : as.i \rangle = \#bs \rangle \rangle$$
{Rango unitario}
$$f.[\ ] = \langle \exists\ bs : bs = [\ ] : \langle \Sigma\ i : 0 \leq i < \#[\ ] : [\ ].i \rangle = \#bs \rangle \rangle$$
{Rango unitario}
$$f.[\ ] = \langle \Sigma\ i : 0 \leq i < \#[\ ] : [\ ].i \rangle = \#[\ ] \rangle$$
{Definición de cardinal y aritmética}
$$f.[\ ] = \langle \Sigma\ i : False : [\ ].i \rangle = \#[\ ] \rangle$$
{Rango vacío}
$$f.[\ ] = True$$

f.(x.xs) = < ∃ as, bs : (x.xs) = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > = #bs >

{Tercero excluido}

f.(x.xs) = < ∃ as, bs : (as ≠ [ ] v as = [ ]) ^ (x.xs) = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > = #bs >

{Distributividad y partición de rango}

f.(x.xs) = < ∃ as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > = #bs > v
< ∃ as, bs : as = [ ] ^ (x.xs) = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > = #bs >

{Reemplazo de as ←a.as y eliminacion de variable con as = [ ]}

f.(x.xs) = < ∃ a, as, bs : (a.as) ≠ [ ] ^ (x.xs) = (a.as) ++ bs : < Σ i : 0 ≤ i < #(a.as) : (a.as).i > = #bs > v
< ∃ bs : (x.xs) = [ ] ++ bs : < Σ i : 0 ≤ i < #[ ] : [ ].i > = #bs >

{Propiedad de listas}

f.(x.xs) = < ∃ a, as, bs : True ^ (x.xs) = a.(as ++ bs) : < Σ i : 0 ≤ i < #(a.as) : (a.as).i > = #bs > v < ∃ bs : (x.xs) = bs : < Σ i : 0 ≤ i < #[ ] : [ ].i > = #bs >

{Neutro de la conjunción, propiedad de listas y definición de cardinal}

f.(x.xs) = < ∃ a, as, bs : (x = a) ^ xs = as ++ bs : < Σ i : 0 ≤ i < 1 + #as : (a.as).i > = #bs > v < ∃ bs : (x.xs) = bs : < Σ i : 0 ≤ i < 0 : [ ].i > = #bs >

{Eliminación de variable con x = a}

f.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < 1 + #as : (x.as).i > = #bs > v < ∃ bs : (x.xs) = bs : < Σ i : 0 ≤ i < 0 : [ ].i > = #bs >

{Aritmética en el rango de la sumatoria, rango unitario}

f.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : i = 0 v 1 ≤ i < 1 + #as : (x.as).i > = #bs > v < Σ i : 0 ≤ i < 0 : [ ].i > = #(x.xs)

{Partición de rango, aritmética en el rango}

f.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : 1 ≤ i < 1 + #as : (x.as).i > + < Σ i : i = 0 : (x.as).i > = #bs > v < Σ i : False : [ ].i > = #(x.xs)

{Cambio de variable i ← i + 1, rango unitario, rango vacío y definición de cardinal}

f.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : 1 ≤ i + 1 < 1 + #as : (x.as).i + 1 > + (x.as).0 = #bs > v (0 = 1 + #xs)

{Aritmética en el rango de la sumatoria, definición de indexación, y aritmética}

f.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + x > = #bs > v False

{Elemento neutro del v}

f.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + x = #bs >

{No puedo aplicar hipótesis inductiva, es necesario generalizar}

genf.k.xs = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + k = #bs >

Probemos que f.xs es un caso particular de la nueva función generalizada :

genf.k.xs = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + k = #bs >

{Elijo k ←0 por ser el elemento neutro de la suma}

genf.0.xs = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + 0 = #bs >

{Aritmética}

genf.0.xs = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > = #bs >

{Definición de f.xs}
genf.0.xs = f.xs

Luego derivemos genf.k.xs mediante inducción en xs.

genf.k.[ ] = < ∃ as, bs : [ ] = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + k = #bs >
{Pasos análogos al caso base anterior}
…
{...}
k = 0

genf.k.(x.xs) = < ∃ as, bs : (x.xs) = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + k = #bs >
{Tercero excluido y partición de rango}
genf.k.(x.xs) = < ∃ as, bs : as ≠ [ ] ^ (x.xs) = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + k = #bs > v < ∃ as, bs : as = [ ] ^ (x.xs) = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + k = #bs >
{Reemplazo de as ←a.as y eliminacion de variable con as = [ ]}
genf.k.(x.xs) = < ∃ a, as, bs : (a.as) ≠ [ ] ^ (x.xs) = (a.as) ++ bs : < Σ i : 0 ≤ i < #(a.as) : (a.as).i > + k = #bs > v < ∃ bs : (x.xs) = [ ] ++ bs : < Σ i : 0 ≤ i < #[ ] : [ ].i > + k = #bs >
{Propiedad de listas, definición de cardinal}
genf.k.(x.xs) = < ∃ a, as, bs : True ^ (x = a) ^ xs = as ++ bs : < Σ i : 0 ≤ i < 1 + #as : (a.as).i > + k = #bs > v < ∃ bs : (x.xs) = bs : < Σ i : 0 ≤ i < 0 : [ ].i > + k = #bs >
{Elemento neutro de la conjunción, eliminación de variable con x = a y rango unitario}
genf.k.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < 1 + #as : (x.as).i > + k = #bs > v < Σ i : 0 ≤ i < 0 : [ ].i > + k = #(x.xs)
{Aritmética en el rango de la sumatoria, aritmética y rango vacío, definición de cardinal}
genf.k.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : i = 0 v 1 ≤ i < 1 + #as : (x.as).i > + k = #bs > v (0 + k = 1 + #xs)
{Partición de rango, rango unitario y cambio de variable i ←i + 1, aritmética}
genf.k.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : 1 ≤ i + 1 < 1 + #as : (x.as).i + 1 > + (x.as).0 + k = #bs > v (k = 1 + #xs)
{Aritmética en el rango, definición de indexación}
genf.k.(x.xs) = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < #as : as.i > + (x + k) = #bs > v (k = 1 + #xs)
{Hipótesis inductiva}
genf.k.(x.xs) = genf.(k + x).xs v (k = 1 + #xs)

Programa final, anotado :
f.[ ] = True
f.(x.xs) = genf.0.xs

$genf.k.[\,] = (k = 0)$
$genf.k.(x.xs) = genf.(k + x).xs \vee (k = (1 + \#xs))$

- $f.xs.n = \langle \exists\ as, bs : xs = as ++ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$

Derivemos f mediante inducción sobre el parámetro xs, con el caso base xs = [ ] y el caso inductivo xs = (x.xs).

$f.[\,].n = \langle \exists\ as, bs : [\,] = as ++ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$
{Propiedad de listas}
$f.[\,].n = \langle \exists\ as, bs : as = [\,] \wedge bs = [\,] : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$
{Anidado}
$f.[\,].n = \langle \exists\ as : as = [\,] : \langle \exists\ bs = [\,] : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle \rangle$
{Rango unitario}
$\langle \exists\ bs = [\,] : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$
{Rango unitario}
$\langle \Sigma\ i : 0 \leq i < \#[\,] : [\,]\,!i * (n - i) \rangle = 8$
{Definición de cardinal}
$\langle \Sigma\ i : 0 \leq i < 0 : [\,]\,!i * (n - i) \rangle = 8$
{Aritmética y rango vacío}
$0 = 8$
{Aritmética}
False

$f.(x.xs).n = \langle \exists\ as, bs : (x.xs) = as ++ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$
{Tercero excluido}
$f.(x.xs).n = \langle \exists\ as, bs : (as \neq [\,] \vee as = [\,]) \wedge (x.xs) = as ++ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$
{Distributividad y partición de rango}
$f.(x.xs).n = \langle \exists\ as, bs : as \neq [\,] \wedge (x.xs) = as ++ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle \vee$
$\langle \exists\ as, bs : as = [\,] \wedge (x.xs) = as ++ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$
{Reemplazo de as $\leftarrow$a.as, eliminación de variable con as = [ ]}
$f.(x.xs).n = \langle \exists\ a, as, bs : (a.as) \neq [\,] \wedge (x.xs) = (a.as) ++ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle \vee$
$\langle \exists\ as, bs : (x.xs) = [\,] ++ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$
{Propiedad de listas}
$f.(x.xs).n = \langle \exists\ a, as, bs : True \wedge (x.xs) = a.(as ++ bs) : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle \vee$
$\langle \exists\ bs : (x.xs) = bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$
{Elemento neutro de la conjunción, propiedad de listas}
$f.(x.xs).n = \langle \exists\ a, as, bs : (x = a) \wedge xs = as ++ bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle \vee$
$\langle \exists\ bs : (x.xs) = bs : \langle \Sigma\ i : 0 \leq i < \#bs : bs\,!i * (n - i) \rangle = 8 \rangle$

{Rango unitario}
f.(x.xs).n = < ∃ a,as, bs : (x = a) ^ xs = as ++ bs : < Σ i : 0 ≤ i < #bs : bs !i ∗ (n − i) > = 8 > v
< Σ i : 0 ≤ i < #(x.xs) : (x.xs) !i ∗ (n − i) > = 8
{Eliminación de variable con x = a y definición de cardinal}
f.(x.xs).n = < ∃ as, bs : xs = as ++ bs : < Σ i : 0 ≤ i < #bs : bs !i ∗ (n − i) > = 8 > v
< Σ i : 0 ≤ i < 1 + #xs : (x.xs) !i ∗ (n − i) > = 8
{Hipótesis inductiva, aritmética en el rango}
f.(x.xs).n = f.xs.n v < Σ i : i = 0 v 1 ≤ i < 1 + #xs : (x.xs) !i ∗ (n − i) > = 8
{Partición de rango}
f.(x.xs).n = f.xs.n v < Σ i : 1 ≤ i < 1 + #xs : (x.xs) !i ∗ (n − i) >
+ < Σ i : i = 0 : (x.xs) !i ∗ (n − i) > = 8
{Cambio de variable i← i + 1 y rango unitario}
f.(x.xs).n = f.xs.n v < Σ i : 1 ≤ i + 1 < 1 + #xs : (x.xs) !i + 1 ∗ (n − i + 1) >
+ (x.xs) !0 ∗ (n − 0) = 8
{Aritmética en el rango, definición de indexación, aritmética}
f.(x.xs).n = f.xs.n v (< Σ i : 0 ≤ i < #xs : xs!i ∗ (n − i + 1) > + (x ∗ n) = 8)
{Tengo que modularizar mod1.xs.n = < Σ i : 0 ≤ i < #xs : xs!i ∗ (n − i + 1) >}
f.(x.xs).n = f.xs.n v (mod1.xs.n + (x ∗ n) = 8)

Derivemos mod1.xs.n mediante inducción en xs :

mod1.[ ].n = < Σ i : 0 ≤ i < #[ ] : [ ]!i ∗ (n − i + 1) >
{Aritmética y rango vacío}
mod1.[ ].n = 0

mod1.(x.xs).n = < Σ i : 0 ≤ i < #(x.xs) : (x.xs)!i ∗ (n − i + 1) >
{Definición de cardinal de lista, aritmética y partición de rango}
mod1.(x.xs).n = < Σ i : 1 ≤ i < 1 + #xs : (x.xs)!i ∗ (n − i + 1) > +
< Σ i : i = 0 : (x.xs)!i ∗ (n − i + 1) >
{Cambio de variable i ← i + 1, rango unitario]
mod1.(x.xs).n = < Σ i : 1 ≤ i + 1 < 1 + #xs : (x.xs)!i + 1 ∗ (n − (i + 1) + 1) > + (x.xs)!0 ∗ (n − 0 + 1)
{Aritmética, definición de indexación}
mod1.(x.xs).n = < Σ i : 0 ≤ i < #xs : xs!i ∗ (n − (i + 1) + 1) > + (x ∗ (n + 1))
{No puedo aplicar hipótesis inductiva, generalizo}
genmod1.xs.n.k = < Σ i : 0 ≤ i < #xs : xs!i ∗ (n − (i + k) + 1) >

Veamos que mod1 es un caso particular de mi función generalizada ya que genmod1.xs.n.0 = mod1.xs.n

genmod1.[ ].n.k = < Σ i : 0 ≤ i < #[ ] : [ ]!i ∗ (n − (i + k) + 1) >
{Definición de cardinal, aritmética y rango vacío}
genmod1.[ ].n.k = 0

$$genmod1.(x.xs).n.k = \langle \Sigma\, i : 0 \leq i < \#(x.xs) : (x.xs)!i * (n - (i + k) + 1) \rangle$$
{Aritmética en el rango}
$$genmod1.(x.xs).n.k = \langle \Sigma\, i : i = 0 \vee 1 \leq i < 1 + \#xs : (x.xs)!i * (n - (i + k) + 1) \rangle$$
{Partición de rango, rango unitario, cambio de variable $i \leftarrow i + 1$}
$$genmod1.(x.xs).n.k = \langle \Sigma\, i : 1 \leq i + 1 < 1 + \#xs : (x.xs)!i + 1 * (n - (i + 1 + k) + 1) \rangle + (x.xs)!0 * (n - (0 + k) + 1)$$
{Aritmética en el rango, definición de cardinal de lista}
$$genmod1.(x.xs).n.k = \langle \Sigma\, i : 0 \leq i < \#xs : xs!i * (n - (i + 1 + k) + 1) \rangle + x. * (n - k + 1)$$
{Hipótesis inductiva}
$$genmod1.(x.xs).n.k = genmod1.xs.n.(k + 1) + x. * (n - k + 1)$$

Programa final, anotado :
f.[ ].n = False
f.(x.xs).n = f.xs.n v (mod1.xs.n + (x ∗ n) = 8)
mod1.xs.n = genmod1.xs.n.0
genmod1.[ ].n.k = 0
genmod1.(x.xs).n.k = genmod1.xs.n.(k + 1) + x. ∗ (n − k + 1)

- $f.xs = \langle \Sigma\, i, j : 0 \leq i \leq j < \#xs : xs.i * xs.j \rangle$

Derivemos la función f mediante inducción en el unico parametro que recibe, con el caso base xs = [ ] y el caso inductivo xs = (x.xs) :

$$f.[\,] = \langle \Sigma\, i, j : 0 \leq i \leq j < \#[\,] : [\,].i * [\,].j \rangle$$
{Definición de cardinal de lista}
$$f.[\,] = \langle \Sigma\, i, j : 0 \leq i \leq j < 0 : [\,].i * [\,].j \rangle$$
{Aritmética}
$$f.[\,] = \langle \Sigma\, i, j : False : [\,].i * [\,].j \rangle$$
{Rango vacío}
$$f.[\,] = 0$$

$$f.(x.xs) = \langle \Sigma\, i, j : 0 \leq i \leq j < \#(x.xs) : (x.xs).i * (x.xs).j \rangle$$
{Definición de cardinal de lista}
$$f.(x.xs) = \langle \Sigma\, i, j : 0 \leq i \leq j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle$$
{Aritmética en el rango}
$$f.(x.xs) = \langle \Sigma\, i, j : (i = 0 \vee 1 \leq i) \wedge i \leq j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle$$
{Distributividad y partición de rango}
$$f.(x.xs) = \langle \Sigma\, i, j : 1 \leq i \wedge i \leq j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle + \langle \Sigma\, i, j : i = 0 \wedge i \leq j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle$$
{Aritmética en el primer rango, eliminación de variable en el segundo}
$$f.(x.xs) = \langle \Sigma\, i, j : 1 \leq i \leq j < 1 + \#xs : (x.xs).i * (x.xs).j \rangle + \langle \Sigma\, j : 0 \leq j < 1 + \#xs : (x.xs).0 * (x.xs).j \rangle$$
{Cambio de variable $i \leftarrow i + 1$, $j \leftarrow j + 1$, definición de indexación}

$$f.(x.xs) = \langle \Sigma\, i, j : 1 \leq i + 1 \leq j + 1 < 1 + \#xs : (x.xs).i + 1 * (x.xs).j + 1 \rangle +$$
$$\langle \Sigma\ \ j :\ 0 \leq j < 1 + \#xs : x * (x.xs).j \rangle$$
{Aritmética en el rango, definición de indexación}
$$f.(x.xs) = \langle \Sigma\, i, j : 0 \leq i \leq j < \#xs : xs.i * xs.j \rangle +$$
$$\langle \Sigma\ \ j :\ 0 \leq j < 1 + \#xs : x * (x.xs).j \rangle$$
{Hipótesis inductiva, aritmética en el segundo rango}
$$f.(x.xs) = f.xs + \langle \Sigma\ \ j : j = 0 \lor 1 \leq j < 1 + \#xs : x * (x.xs).j \rangle$$
{Partición de rango}
$$f.(x.xs) = f.xs + \langle \Sigma\ \ j : 1 \leq j < 1 + \#xs : x * (x.xs).j \rangle + \langle \Sigma\ \ j : j = 0 : x * (x.xs).j \rangle$$
{Cambio de variable j ←j + 1, rango unitario}
$$f.(x.xs) = f.xs + \langle \Sigma\ \ j : 1 \leq j + 1 < 1 + \#xs : x * (x.xs).j + 1 \rangle + x * (x.xs).0$$
{Aritmética en el rango, definición de indexación}
$$f.(x.xs) = f.xs + \langle \Sigma\, j : 0 \leq j < \#xs : x * xs.j \rangle + (x * x)$$
{Modularizo $sum\_mult.k.xs = \langle \Sigma\, j : 0 \leq j < \#xs : k * xs.j \rangle$}

Derivemos sum_mult.k.xs repitiendo los mismos casos que en la anterior derivación :

$$sum\_mult.k.[\,] = \langle \Sigma\, j : 0 \leq j < \#[\,] : x * [\,].j \rangle$$
{Definición de cardinal de lista}
$$sum\_mult.k.[\,] = \langle \Sigma\, j : 0 \leq j < 0 : x * [\,].j \rangle$$
{Aritmética y rango vacío}
$$sum\_mult.k.[\,] = 0$$

$$sum\_mult.k.(x.xs) = \langle \Sigma\, j : 0 \leq j < \#(x.xs) : k * (x.xs).j \rangle$$
{Definición de cardinal de lista}
$$sum\_mult.k.(x.xs) = \langle \Sigma\, j : 0 \leq j < 1 + \#xs : k * (x.xs).j \rangle$$
{Aritmética en el rango}
$$sum\_mult.k.(x.xs) = \langle \Sigma\, j : j = 0 \lor 1 \leq j < 1 + \#xs : k * (x.xs).j \rangle$$
{Partición de rango}
$$sum\_mult.k.(x.xs) = \langle \Sigma\, j : 1 \leq j < 1 + \#xs : k * (x.xs).j \rangle + \langle \Sigma\, j : j = 0 : k * (x.xs).j \rangle$$
{Cambio de variable j ←j + 1, rango unitario}
$$sum\_mult.k.(x.xs) = \langle \Sigma\, j : 1 \leq j + 1 < 1 + \#xs : k * (x.xs).j + 1 \rangle + (k * (x.xs).0)$$
{Aritmética en el rango, definición de indexación}
$$sum\_mult.k.(x.xs) = \langle \Sigma\, j : 0 \leq j < \#xs : k * xs.j \rangle + (k * x)$$
{Hipótesis inductiva}
$$sum\_mult.k.(x.xs) = sum\_mult.k.xs + (k * x)$$

Programa final, anotado :
f. [ ] = 0
f.(x.xs) = f.xs + sum_mult.k.xs + (x ∗ x)
sum_mult.k.[ ] = 0
sum_mult.k.(x.xs) = sum_mult.k.xs + (k ∗ x)

Especificaciones y corrida de ejemplos en el paradigma Funcional

1. f.xs = “ La suma de los elementos impares de xs es par “.

Propuesta para la especificación = par.(< Σ i : 0 <= i < #xs ^ impar.i : xs!i >). En donde par.n = ((n mod 2) = 0).

- El tipo de la función es Bool, ya que recibe un entero y determina en este caso si es par o no.
- Propuesta de función mediante recursión :

f. [ ] = True
f. (x.xs) = par.(sumImpares.x) ^ f.xs

- as tal que f. as = True. Propongo as = [0, 2, 1, 4, 3, 8]
- bs tal que f. bs = False. Propongo bs = [0, 3, 2, 9, 4, 23]
- Evaluar f. [2, 4, 5, 8] según la especificación.

f.xs = par.(< Σ i : 0 <= i < #xs ^ impar.i : xs!i >)
{Especificación sabiendo que #xs = 4}
f. [2, 4, 5, 8] = par.(< Σ i : i e {0, 1, 2, 3} ^ impar.i : xs!i >)
{Caso impar.i = True}
f. [2, 4, 5, 8] = par.(< Σ i : i e {1, 3} : xs!i >)
{Definición de sum}
f. [2, 4, 5, 8] = par.(T(1) + T(3))
{T = xs!i}
f. [2, 4, 5, 8] = par.(xs!1 + xs!3)
{Indexación}
f. [2, 4, 5, 8] = par.(4 + 8)
{Aritmética}
f. [2, 4, 5, 8] = par.(12)
{Definición de par}
f. [2, 4, 5, 8] = True

2. f.xs = “Hay un elemento impar en xs”.

Propuesta para la especificación f.xs = < ∃ i : 0 <= i <= #xs : impar.(xs ! i) >. En donde impar.n = ((n mod 2) = 1).

- El tipo de f.xs es claramente Bool, ya que determina si es verdadera o no la existencia de un elemento impar en xs.
- as tal que #as > 2 y f. as = False. Propongo as = [2, 4, 1, 8, 10].
- Evaluar f. as

f.xs = < ∃ i : 0 <= i <= #xs : impar.(xs ! i) >
{Especificación sabiendo que #xs = 5}
f.[2, 4, 1, 8, 10] = < ∃ i : i e {0, 1, 2 ,3, 4} : impar.(xs ! i) >
{Aplicamos el término a los elementos del rango}
f.[2, 4, 1, 8, 10] = T(0) v T(1) v T(2) v T(3) v T(4)
{T = impar.(xs ! i)}

f.[2, 4, 1, 8, 10] = impar.(xs ! 0) v impar.(xs ! 1) v impar.(xs ! 2) v impar.(xs ! 3) v impar.(xs ! 4)

{Definición de indexación}

f.[2, 4, 1, 8, 10] = impar.(2) v impar.(4) v impar.(1) v impar.(8) v impar.(10)

{Definición de impar}

f.[2, 4, 1, 8, 10] = False v False v True v False v False

{Aritmética}

f.[2, 4, 1, 8, 10] = True

Derivaciones en el paradigma Imperativo

Const M : Int;
    a : array [0,M) of Int;
Var r : Bool;
$\{M => 0\}$
S
$\{r = < \forall\ i : 0 <= i < M : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >>\}$

Comencemos planteando el invariante, como mi post condición no es una conjunción, propongo un invariante I mediante la técnica de reemplazo de constante por variable.
$I = \{r = < \forall\ i : 0 <= i < m : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >> \wedge$
$0 <= m <= M\}$
Ahora toca ver, $P \rightarrow I$, que es claro que no, pues M => 0 no es lo suficientemente fuerte para implicar al invariante, por lo tanto, propongo una inicialización

$$P \rightarrow wp\ (r, m := E, F)\ (I)$$
{Definición de wp para asignaciones}
$$\{E = < \forall\ i : 0 <= i < F : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >> \wedge$$
$$0 <= F <= M\}$$
{Propongo $F \leftarrow 0$}
$$\{E = < \forall\ i : 0 <= i < 0 : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >> \wedge$$
$$0 <= 0 <= M\}$$
{Suponiendo que M => 0}
$$\{E = < \forall\ i : 0 <= i < 0 : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >> \wedge True\}$$
{Aritmética en el rango, neutro de ^}
$$\{E = < \forall\ i : False : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >>\}$$
{Rango vacío}
$$E = True$$
{Propongo $E \leftarrow True$}
$$True = True$$
{Reflexividad}
$$True$$

Veamos ahora qué $I \wedge \neg B \rightarrow Q$, esto suponiendo que el antecedente y comprobando el consecuente. Introducir la guarda $B = m \neq M$, ya veremos más adelante el porqué de esta decisión.

$$\{r = < \forall\ i : 0 <= i < m : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >> \wedge$$
$$0 <= m <= M\} \wedge m = M \rightarrow \{r = < \forall\ i : 0 <= i < M : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >>\}$$
{Suponiendo que m <= M y m = M, puedo deducir que m = M}
$$\{r = < \forall\ i : 0 <= i < m : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >> \} \wedge m = M$$
$$\rightarrow \{r = < \forall\ i : 0 <= i < M : < \Sigma j : 0 <= j <= i : a.j > => < N\ j : 0 <= j < i : a.j => 0 >>\}$$
{Leibniz}

$$\{r = < \forall\ i : 0 <= i < M : < \Sigma j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \}\ \{r = < \forall\ i : 0 <= i < M : < \Sigma j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \}$$

$$\{P \rightarrow P = True\}$$

$$True$$

En este punto es hora de ver una función t llamada cota, tal que sea positiva, propongo t = M - m, pues mi guarda B me indica que m nunca llegará a ser igual a M por lo tanto su diferencia es positiva.

Ahora veamos que {I ^ B} S { I },es decir, busquemos el cuerpo del bucle, para eso propongo una asignación (r, m := E, m + 1) y lo pruebo mediante wp.

$$wp\ (r, m := E, m + 1)\ (\ I\ )$$

$$\{\text{Definición de wp para asignaciones})$$

$$\{E = < \forall\ i : 0 <= i < m + 1 : < \Sigma j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \wedge\ 0 <= m + 1 <= M\}$$

$$\{\text{Aritmética en el rango}\}$$

$$\{E = < \forall\ i : i = m \vee 0 <= i < m : < \Sigma j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \wedge\ 0 <= m + 1 <= M\}$$

$$\{\text{Partición de rango y suponiendo que } (m <= M) \wedge (m \neq M) \rightarrow (m < M) \rightarrow (m + 1) <= M\}$$

$$\{E = < \forall\ i : i = m : < \Sigma j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \wedge\ < \forall\ i : 0 <= i < m : < \Sigma j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \wedge\ True\}$$

$$\{\text{Absorbente de } \wedge\text{, Rango unitario en el primer término}\}$$

$$\{E = < \Sigma j : 0 <= j <= m : a.j > \Rightarrow < N\ j : 0 <= j < m : a.j \Rightarrow 0 > \wedge\ < \forall\ i : 0 <= i < m : < \Sigma j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \}$$

$$\{\text{Suposición de r}\}$$

$$\{E = < \Sigma j : 0 <= j <= m : a.j > \Rightarrow < N\ j : 0 <= j < m : a.j \Rightarrow 0 > \wedge\ r\ \}$$

$$\{\text{Fortalecimiento de invariante con la introducción de la variable u y k}\}$$

Nuevo Invariante I = $\{r = < \forall\ i : 0 <= i < m : < \Sigma j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \wedge (0 <= m <= M) \wedge (u = < \Sigma j : 0 <= j <= m : a.j >) \wedge (k = < N\ j : 0 <= j < m : a.j \Rightarrow 0 > \}$

Volvamos a probar todo de nuevo, ahora con nuestro invariante fortalecido.
Para ver la inicialización, P → I, volvemos a lo antes propuesto y vemos como quedaría u y k.

$$P \rightarrow wp\ (r, m, u, k := E, F, G, H)\ (I)$$

$$\{\text{Definición de wp para asignaciones}\}$$

$$\{E = < \forall\ i : 0 <= i < F : < \Sigma\ j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \wedge (0 <= F <= M) \wedge (G = < \Sigma j : 0 <= j <= F : a.j >) \wedge (H = < N\ j : 0 <= j < F : a.j \Rightarrow 0 > \}$$

$$\{\text{Propongo } F \leftarrow 0\}$$

$$\{E = < \forall\ i : 0 <= i < 0 : < \Sigma\ j : 0 <= j <= i : a.j > \Rightarrow < N\ j : 0 <= j < i : a.j \Rightarrow 0 >> \wedge (0 <= 0 <= M) \wedge (G = < \Sigma\ j : 0 <= j <= 0 : a.j >) \wedge (H = < N\ j : 0 <= j < 0 : a.j \Rightarrow 0 > \}$$

$$\{\text{Aritmética}\}$$

{E = < ∀ i : False : < Σj : 0 <= j <= i : a.j > => < N j : 0<= j < i : a.j => 0 >> ^ True ^ (G = < Σ j : j = 0 : a.j >) ^ (H = < N j : False : a.j => 0 > }

{Rango Vacío dos veces, y rango unitario, además neutro de ^}

{E = True ^ True ^ G = a.0 ^ H = 0}

{Aritmética}

{E = True ^ G = a.0 ^ H = 0}

{E ← True, G ← a.0, H ← 0}

{True = True ^ a.0 = a.0 ^ 0 = 0}

{Reflexividad}

True

Ahora al ver que I ^ ¬B → Q, como nuestro invariante es más fuerte, es obvio que también se cumple, por lo que es trivial.

La cota t, sigue siendo t = M - m.

Veamos ahora el cuerpo del ciclo, proponiendo una asignación a las variables correspondientes.

{I ^ B} S { I }

wp (r, m, u, k := E, m + 1, F, G) (I)

{Definición de wp}

{E = < ∀ i : 0 <= i < m + 1 : < Σj : 0 <= j <= i : a.j > => < N j : 0<= j < i : a.j => 0 >> ^ (0 <= m + 1 <= M) ^ (u = < Σj : 0 <= j <= m + 1 : a.j >) ^ (k =< N j : 0 <= j < m + 1 : a.j => 0 > }

{Suponiendo que m <= M y m distinto de M → m + 1 <= M}

{E = < ∀ i : 0 <= i < m + 1 : < Σj : 0 <= j <= i : a.j > => < N j : 0<= j < i : a.j => 0 >> ^ True ^ (u = < Σj : 0 <= j <= m + 1 : a.j >) ^ (k =< N j : 0 <= j < m + 1 : a.j => 0 > }

{Absorbente de ^}

{E = < ∀ i : 0 <= i < m + 1 : < Σj : 0 <= j <= i : a.j > => < N j : 0<= j < i : a.j => 0 >> ^ (u = < Σj : 0 <= j <= m + 1 : a.j >) ^ (k =< N j : 0 <= j < m + 1 : a.j => 0 > }

{Aritmética en el rango del primer término}

{E = < ∀ i : i = m v 0 <= i < m : < Σj : 0 <= j <= i : a.j > => < N j : 0<= j < i : a.j => 0 >> ^ (u = < Σj : 0 <= j <= m + 1 : a.j >) ^ (k =< N j : 0 <= j < m + 1 : a.j => 0 > }

{Partición de rango}

{E = < ∀ i : i = m : < Σj : 0 <= j <= i : a.j > => < N j : 0<= j < i : a.j => 0 >> ^ < ∀ i : 0 <= i < m : < Σj : 0 <= j <= i : a.j > => < N j : 0<= j < i : a.j => 0 >> ^ (u = < Σj : 0 <= j <= m + 1 : a.j >) ^ (k =< N j : 0 <= j < m + 1 : a.j => 0 > }

{Suposición de r}

{E = < ∀ i : i = m : < Σj : 0 <= j <= i : a.j > => < N j : 0<= j < i : a.j => 0 >> ^ r ^ (u = < Σj : 0 <= j <= m + 1 : a.j >) ^ (k =< N j : 0 <= j < m + 1 : a.j => 0 > }

{Rango unitario}

{E = (< Σj : 0 <= j <= m : a.j > => < N j : 0 <= j < m : a.j => 0 > ^ r) ^ (u = < Σj : 0 <= j <= m + 1 : a.j >) ^ (k =< N j : 0 <= j < m + 1 : a.j => 0 > }

{Suposición de u y k}

{E = (u => k ^ r) ^ (u = < Σj : 0 <= j <= m + 1 : a.j >) ^ (k =< N j : 0 <= j < m + 1 : a.j => 0 > }

{Aritmética en el rango de k}
{E = (u => k ^ r) ^ (u = < Σj :0 <= j <= m + 1 : a.j >) ^ (k = < N j : j = m v 0 <= j < m : a.j => 0 > }
{Partición de rango}
{E = (u => k ^ r) ^ (u = < Σj :0 <= j <= m + 1 : a.j >) ^ (k = < N j : j = m : a.j => 0 > + < N j : 0 <= j < m : a.j => 0 > }
{Rango unitario, suposición de k}
{E = (u => k ^ r) ^ (u = < Σj :0 <= j <= m + 1 : a.j >) ^ (k = (a.m => 0) + k }
{Aritmética}
{E = (u => k ^ r) ^ (u = < Σj : j = m + 1 v 0 <= j <= m : a.j >) ^ (k = (a.m => 0) + k }
{Separación de un término}
{E = (u => k ^ r) ^ (u = a.(m+1) + < Σj : 0 <= j <= m : a.j >) ^ (k = (a.m => 0) + k }
{Suposición de u}
{E = (u => k ^ r) ^ (u = a.(m+1) + u) ^ (k = (a.m => 0) + k) }
{E ← (u => k ^ r), F ← (u = a.(m+1) + u), G ← (k = (a.m => 0) + k)}
{Reflexividad}
True

Probemos ahora que la cota t, es positiva.

I ^ B → t => 0
{Especificación}
I ^ B → (M - m) => 0
{Suposición m <= M ^ m ≠ M → m < M}
True

Por último, veamos que la cota t es decreciente, que a priori es claro ya que incrementamos a m por lo que M es cada vez menor.

{I ∧ B ∧ t = T}
S
{t < T}
t < M - m
{Especificación}
M - (m + 1) < M - m
{Aritmética}
M -m - 1 < M - m
{Más aritmética}
-1 < 0
{Es claro que - 1 es menor que 0}
True

Finalmente hemos derivado el programa, y queda de la pinta:

```
Const M : Int;
      a : array [0,M) of Int;
Var r : Bool;
{M => 0}
r ,m, u, k := True, 0, a.0,  0;
```

{ I }
do (m ≠ M)
{ I ^ B }
    r, m, u, k := (u => k ^ r), m +1, (a.(m+1) + u), (k = (a.m => 0) + k);
{ I }
od
{I ^ ¬B}
{r = < ∀ i : 0 <= i < M : < Σj : 0 <= j <= i : a.j > => < N j : 0<= j < i : a.j => 0 >>}

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

Encontremos el invariante, en este caso mediante la técnica de reemplazo de constante por variable : Inv = r = < N i, j : 0 ≤ i < j < n : A.i = A.j > ^ 0 <= n <= N con la guarda B = n ≠ N o bien n <= N.

- Ahora veamos que claramente no se cumple que “P → I” ya que mi precondición es muy débil. Por ello debemos encontrar una inicialización.

Propongo entonces

P → wp(n,r := 0, E) (I)
{Definición de wp para asignación}
E = < N i, j : 0 ≤ i < j < 0 : A.i = A.j > ^ 0 <= 0 <= N
{Por mi precondición se que vale 0 <= N}
E = < N i, j : 0 ≤ i < j < 0 : A.i = A.j > ^ True
{Absorbente de la conjunción}
E = < N i, j : 0 ≤ i < j < 0 : A.i = A.j >
{Lógica en el rango}
E = < N i, j : False : A.i = A.j >
{Rango vacío}
E = 0
{Elijo E = 0}
0 = 0
True

Encontré mi inicialización, S0 = E,r := 0, 0.

- Ahora deberíamos de probar que I ^ ¬B → Q (ya garantizado por la elección del invariante). Asumimos el invariante y la n = N, y queda trivial.

- Probemos ahora el cuerpo del ciclo tal que {I ^ B} S {I} de forma que una vez que el ciclo se realice y termine, el I siga valiendo.

{I ^ B} → wp(r,n := E,F) (I)

{Definición de wp de asignación}

$E = \langle N\ i, j : 0 \leq i < j < F : A.i = A.j \rangle \wedge 0 <= F <= N$

{Propongo F := n + 1}

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

{El lado derecho vale por que tengo como hipótesis 0 <= n y n < N}

$E = \langle N\ i, j : 0 \leq i < j < n + 1 : A.i = A.j \rangle \wedge True$

{Absorbente de la conjunción y aritmética en el rango}

$E = \langle N\ i, j : 0 \leq i < j \wedge (j = n \vee j < n) : A.i = A.j \rangle$

{Partición de rango}

$E = \langle N\ i, j : 0 \leq i < j < n \wedge j = n : A.i = A.j \rangle + \langle N\ i, j : 0 \leq i < j < n \wedge j < n : A.i = A.j \rangle$

{Eliminación de variable en el primer término, junto la desigualdad en el segundo término}

$E = \langle N\ i: 0 \leq i < n : A.i = A.n \rangle + \langle N\ i, j : 0 \leq i < j < n : A.i = A.j \rangle$

{Suposición sobre r en el segundo término}

$E = \langle N\ i: 0 \leq i < n : A.i = A.n \rangle + r$

{Tenemos problema de bordes, propongo un ciclo anidado}

Notemos que debería de tener una nueva variable debido a que tengo una parte que no es programable. Introduzco s = < N i: 0 ≤ i < n : A.i = A.n >. Ahora, tenemos algo de la pinta:

{v ^ B}
S3;
{Inv ^ B ^ s = < N i: 0 ≤ i < n : A.i = A.n >}
r, n := r + s, n + 1;
{Inv}

En "S3" no debo tocar r,n. Por ende, introducimos otro invariante I' tal que Inv' = I ^ B ^ s =< N i: 0 ≤ i < m : A.i = A.n> ^ 0 <= m <= n. Y la otra guarda B' = m ≠ n.

{Inv ^ B}
S4;
{Inv'}
do m ≠ n →
    {Inv' ^ B'}
    S5;
    {Inv'}
od
{Inv ^ B ^ s = < N i: 0 ≤ i < n : A.i = A.n >}

En donde S4 es la inicialización de mi ciclo anidado, y S5 el cuerpo del bucle.

- P → I

wp(m,s := E, F) (I)

{Definición de wp}

I ^ B ^ F =< N i: 0 ≤ i < E : A.i = A.n> ^ 0 <= E <= n

{Propongo E ← 0}

I ^ B ^ F =< N i: 0 ≤ i < 0 : A.i = A.n> ^ 0 <= 0 <= n
{El I y B valen por hipótesis, así como también 0 <= n}
True ^ F =< N i: 0 ≤ i < 0 : A.i = A.n> ^ True
{Absorbente de la conjunción}
F =< N i: 0 ≤ i < 0 : A.i = A.n>
{Aritmética]
F =< N i: False : A.i = A.n>
{Rango vacío}
F = 0
{Elijo F ← 0}
True

{Inv ^ B}
m,s = 0, 0;
{Inv'}
do m ≠ n →
{Inv' ^ B'}
if A.i = A.n →
m,s := m + 1, s + 1;
A.i ≠ A.m →
m := m +1;
fi
{Inv'}
od
{Inv ^ B ^ s = < N i: 0 ≤ i < n : A.i = A.n >}

Ahora veamos el cuerpo del ciclo "S5"

- {I ^ B} S {I} → wp(m,s := g, h) (I)

wp(m,s := g, h) (I)
{Definición de wp para la asignación}
I ^ B ^ H = < N i: 0 ≤ i < g : A.i = A.n > ^ 0 <= g <= n
{El I y B valen por suposición}
True ^ H = < N i: 0 ≤ i < g : A.i = A.n > ^ 0 <= g <= n
{Propongo g ← m + 1 y neutro conjunción}
H = < N i: 0 ≤ i < m + 1 : A.i = A.n > ^ 0 <= m + 1 <= n
{Por suposición es valido m + 1 <= n}
H = < N i: 0 ≤ i < m + 1 : A.i = A.n > ^ True
{Absorbente de la conjunción}
H = < N i: 0 ≤ i < m + 1 : A.i = A.n >
{Partición de rango}
H = < N i: 0 ≤ i < m : A.i = A.n > + < N i: i = m : A.i = A.n >
{Suposición de s en el primer término}
H = s + < N i: i = m : A.i = A.n >
{Rango unitario del conteo}
H = (A.i = A.m → 1 + s

A.i ≠ A.m → s

)

Ahora debemos ver que ambos ciclos terminen.

- I ^ B → t => 0
- {I ^ B ^ t = T} s { t < T}

t =>0

{Especificación y propuesta de cota T = N - n}

I ^ B → N - n => 0

{Suposición de (n ≠ N) ^ (0 <= n <= N) → n < N}

N - n => 0

True

{I ^ B ^ t = T} s { t < T}

t < N - n

{Especificación de t, asignación de n := n+1}

N - (n + 1) < N - n

{Aritmética}

N - n - 1 < N - n

{Aritmética}

N - 1 < N

True

Trivial con las guardas y la cota del ciclo anidado.

Programa final:

```
Const N : Int,
A : array [0, N) of Int;
Var r, n, m, s : Int
{P : N ≥ 0}
n, r := 0, 0;
do n ≠ N →
    m, s := 0,0;
    do m ≠ n →
        if A .i = A .m →
           m, s := m + 1, s + 1;
           A.i ≠ A.m →
           m := m + 1;
        fi
     od
n, r := n + 1, r + s;
od
{Q : r =  < N i, j : 0 ≤ i < j < N : A.i = A.j > }
```

Const M : Int;
a : array [0,M) of Int;
Var r : Int;
{M => 0}
S
{r = < N i, j : 0 <= i < j < M : par. (a.i + a.j)}

Veamos que S debe ser un ciclo, luego derivemos :
Primero propongamos un Invariante, mediante la técnica de reemplazo de constante por variable, así ya cumplimos el requisito I ^ ¬B → Q :

I = {r = < N i, j : 0 <= i < j < m : par. (a.i + a.j) ^ 0 <= m <= M} y de aquí obtenemos B = m ≠ M.

- Inicialización P → I, claramente mi precondición no implica al invariante, luego necesito de una asignación. Trabajamos con wp o simplemente probamos una asignación. En este caso propongo r := 0 y m := 0.

r = < N i, j : 0 <= i < j < m : par. (a.i + a.j) ^ 0 <= m <= M
{Especificación}
0 = < N i, j : 0 <= i < j < 0 : par. (a.i + a.j) ^ 0 <= 0 <= M
{La parte derecha por suposición es verdadera}
0 = < N i, j : 0 <= i < j < 0 : par. (a.i + a.j) ^ True
{Neutro de la conjunción}
0 = < N i, j : 0 <= i < j < 0 : par. (a.i + a.j) >
{Lógica}
0 = < N i, j : False : par. (a.i + a.j) >
{Rango vacío}
0 = 0
True

S0 →r, m := 0,0.

Ahora veamos el cuerpo del ciclo, S1. Probemos primero con una inicialización y veamos que sale. Primero propongo m := m + 1 y r := E.
{I ^ B} s {I}

wp(m, r := m + 1, E) (I)
{Definición de wp para asignación}
E = < N i, j : 0 <= i < j < m + 1 : par. (a.i + a.j) > ^ 0 <= m + 1 <= M
{Por suposición el término de la derecha es verdadero}
E = < N i, j : 0 <= i < j < m + 1 : par. (a.i + a.j) > ^ True
{Neutro de la conjunción}
E = < N i, j : 0 <= i < j < m + 1 : par. (a.i + a.j) >
{Partición de rango con j < m o j = m y eliminación de j con m }
E = < N i, j : 0 <= i < j < m: par. (a.i + a.j) > + < N i: 0 <= i < m: par. (a.i + a.m) >
{En el lado izquierdo aplicamos suposición}
E = r + < N i: 0 <= i < m: par. (a.i + a.m) >
{Existe problema de borde, por ende propongamos un ciclo anidado}

Lo que antes era S1 (asignación) ahora deja de serlo y es una secuenciación. Debemos introducir una nueva variable “s” tal que: s = < N i: 0 <= i < m: par. (a.i + a.m) >
Notemos que S3 ahora va a tener una inicialización, y un cuerpo, ya que es otro ciclo. Por ende necesitamos otro Invariante I’.
I’ = { I ^ B ^ s = < N i: 0 <= i < k : par. (a.i + a.m) >} con la guarda B = k ≠ m.

{Inv ^ B}
S3;
{Inv ^ B ^ s = < N i: 0 <= i < m: par. (a.i + a.m) > }
r, n := r + s, n + 1;
{Inv}

Refinemos el ciclo tal que:
{Inv ^ B}
S4;
{Inv’}
do k ≠ m →
    {Inv’ ^ B’}
    S5;
    {Inv’}
od
{Inv ^ B ^ s = < N i: 0 <= i < k : par. (a.i + a.m) > }

Toca encontrar S4 y S5. Busquemos la inicialización S4:

- P → I ? Claramente no, necesitamos una asignación.

wp(k, s := 0, 0) (I’)
{Definición de wp para asignación}
Inv ^ B ^ 0 = < N i: 0 <= i < 0 : par. (a.i + a.m) >
{Aritmética}
Inv ^ B ^ 0 = < N i: False : par. (a.i + a.m) >
{Rango vacío}
Inv ^ B ^ 0 = 0
{Por suposición}
True ^ 0 = 0
{Neutro de la conjunción}
0 = 0
True

Ahora encontramos el cuerpo del ciclo.
{Inv’ ^ B’} **s5** {Inv’}

wp(k, s := k + 1, E) (I’)
{Definición de wp para asignación}

Inv ^ B ^ E = < N i: 0 <= i < k + 1 : par. (a.i + a.m) >

{El lado izquierdo es válido por suposición}

True ^ E = < N i: 0 <= i < k + 1 : par. (a.i + a.m) >

{Neutro de la conjunción}

E = < N i: 0 <= i < k + 1 : par. (a.i + a.m) >

{Aritmética en el rango}

E = < N i: 0 <= i < k v i = k : par. (a.i + a.m) >

{Partición de rango}

E = < N i: 0 <= i < k v i = k : par. (a.i + a.m) >

{Partición de rango}

E = < N i: 0 <= i < k : par. (a.i + a.m) > + < N i: i = k : par. (a.i + a.m) >

{Suposición en el lado izquierdo}

E = s + < N i: i = k : par. (a.i + a.m) >

{Rango unitario}

E = (par. (a.i + a.m) → 1 + s
¬par. (a.i + a.m) → s
)

Quedaría por ver que la función cota t es positiva, y que decrece en ambos bucles.
Para ello propongo t = M - m y t' = m - k.
Finalmente, el programa anotado:

```
Const M : Int;
a : array [0,M) of Int;
Var r,k,s,m : Int;
r, m : = 0, 0;
do m ≠ M →
    k, s := 0, 0;
    do k ≠ m →
        if par. (a.i + a.m) →
                k, s := k + 1, 1 + s ;
        ¬par. (a.i + a.m) →
                k, s := k + 1, s ;
        fi
    od
r, m := r + s, m + 1;
od
```

Const N : Int, A : array [0, N) of Int;
Var r : Bool;
{P : N ≥ 0}
S
{Q : r = ⟨ ∃ i, j : 0 ≤ i < j < N : A.i + A.j = 8 ⟩}

Veamos que claramente necesitamos un ciclo para recorrer el arreglo y determinar si existen dos elementos tal que sumados den 8. Luego planteamos el invariante mediante la técnica de reemplazo por constante.
$I = \{r = \langle \exists\ i, j : 0 \leq i < j < n : A.i + A.j = 8 \rangle \wedge 0 <= n <= N\}$, de esta forma, garantizamos que $I \wedge \neg B \rightarrow Q$ (prueba trivial). Además, introducimos la guarda $B = n \neq N$.

- Veamos si $P \rightarrow Q$, de entrada vemos que esto no se cumple, propongo entonces una inicialización S0.

$$wp(n,r := E, G)\ (I)$$
{Definición de wp}
$$G = \langle \exists\ i, j : 0 \leq i < j < E : A.i + A.j = 8 \rangle \wedge 0 <= E <= N$$
{Propongo $E \leftarrow 0$}
$$G = \langle \exists\ i, j : 0 \leq i < j < 0 : A.i + A.j = 8 \rangle \wedge 0 <= 0 <= N$$
{Suposición del lado derecho}
$$G = \langle \exists\ i, j : 0 \leq i < j < 0 : A.i + A.j = 8 \rangle \wedge True$$
{Absorbente de la conjunción y aritmética en el rango}
$$G = \langle \exists\ i, j : False : A.i + A.j = 8 \rangle$$
{Rango vacío}
$$G = False$$
{Elijo $G \leftarrow False$}
$$False = False$$
$$True$$

- Veamos ahora el cuerpo del ciclo. Propongo pensarlo como asignación y ver a que llego, $\{I \wedge B\}\ \{r, n := E, n + 1\}\ \{I\}$.

$$wp(r, n := E, n + 1)\ (I)$$
{Definición de wp}
$$E = \langle \exists\ i, j : 0 \leq i < j < n + 1 : A.i + A.j = 8 \rangle \wedge 0 <= n + 1 <= N$$
{Suposición de $0 <= n <= N$ y $n \neq N$}
$$E = \langle \exists\ i, j : 0 \leq i < j < n + 1 : A.i + A.j = 8 \rangle \wedge True$$
{Neutro de la conjunción y aritmética en el rango}
$$E = \langle \exists\ i, j : 0 \leq i < j \wedge (j = n \vee j < n): A.i + A.j = 8 \rangle$$
{Distributividad y Partición de rango}
$$E = \langle \exists\ i, j : 0 \leq i < j \wedge j = n: A.i + A.j = 8 \rangle \vee \langle \exists\ i, j : 0 \leq i < j \wedge (j < n): A.i + A.j = 8 \rangle$$
{Eliminación de variable en primer término, aritmética en el segundo}
$$E = \langle \exists\ i: 0 \leq i < n: A.i + A.n = 8 \rangle \vee \langle \exists\ i, j : 0 \leq i < j < n: A.i + A.j = 8 \rangle$$
{Suposición en el segundo término}
$$E = \langle \exists\ i: 0 \leq i < n: A.i + A.n = 8 \rangle \vee r$$
{Problema de borde, no puedo reforzar invariante}
Creamos una nueva variable $s = \langle \exists\ i: 0 \leq i < n: A.i + A.n = 8 \rangle$
{Especificación}
$$E = s \vee r$$
{Elijo $E \leftarrow s \vee r$}
$$s \vee r = s \vee r$$
$$True$$

Ahora nos encontramos con que no nos funciona tener una asignación en el cuerpo del ciclo, por ende vemos que es una secuenciación. Por ende planteamos un nuevo invariante I' = {Inv ^ B ^ s = ⟨ ∃ i: 0 ≤ i < k: A.i + A.n = 8 ⟩ ^ 0 <= k <= n} y luego tenemos la segunda guarda B' = k ≠ n.

- Encontremos P' → I' (inicialización del ciclo anidado)

wp(s, k := E,G) (I')
{Definición de wp}
Inv ^ B ^ E = ⟨ ∃ i: 0 ≤ i < G : A.i + A.n = 8 ⟩ ^ 0 <= G <= n
{Propongo G ← 0}
Inv ^ B ^ E = ⟨ ∃ i: 0 ≤ i < 0 : A.i + A.n = 8 ⟩ ^ 0 <= 0 <= n
{En el lado derecho y en el izquierdo aplicamos suposición}
True ^ E = ⟨ ∃ i: 0 ≤ i < 0 : A.i + A.n = 8 ⟩ ^ True
{Neutro de la conjunción}
E = ⟨ ∃ i: 0 ≤ i < 0 : A.i + A.n = 8 ⟩
{Aritmética en el rango y rango vacío}
E = False
{Elijo E ← False}
False = False
True

- Nos queda ahora por encontrar el cuerpo del ciclo anidado tal que : {I' ^ B'} s {I'}

wp(s, k := E, k + 1) (I')
{Definición de wp para asignación}
Inv ^ B^ E = ⟨ ∃ i: 0 ≤ i < k + 1: A.i + A.n = 8 ⟩ ^ 0 <= k + 1 <= n
{Suposición de antecedente y Neutro de conjunción}
E = ⟨ ∃ i: 0 ≤ i < k + 1: A.i + A.n = 8 ⟩ ^ 0 <= k + 1 <= n
{Suposición de antecedente y Neutro de conjunción }
E = ⟨ ∃ i: 0 ≤ i < k + 1: A.i + A.n = 8 ⟩
{Aritmética en el rango}
E = ⟨ ∃ i: 0 ≤ i < k v i = k : A.i + A.n = 8 ⟩
{Distributividad y Partición de rango}
E = ⟨ ∃ i: i = k : A.i + A.n = 8 ⟩ v ⟨ ∃ i: 0 ≤ i ^ i < k : A.i + A.n = 8 ⟩
{rango unitario en el primer cuantificador y aritmética en el segundo}
E = (A.k + A.n = 8) v ⟨ ∃ i: 0 ≤ i < k : A.i + A.n = 8 ⟩
{Suposición}
E = (A.k + A.n = 8) v s
{Elijo E ← (A.k + A.n = 8) v s}
(A.k + A.n = 8) v s = (A.k + A.n = 8) v s
True

Const N : Int, A : array [0, N) of Int;
Var r : Bool;
Var s, n, k := Int;
{P : N ≥ 0}

```
r, n := False, 0;
do n ≠ N →
    s, k := False, 0;
    do k ≠ n →
        s, k := (A.k + A.n = 8)  v s, k + 1;
    od
r, n := s v r , n + 1;
od
```

$\{Q : r = \langle\ \exists\ i, j : 0 \leq i < j < N : A.i + A.j = 8 \rangle\}$

Const M : Int;
Var A : array[0, M) of Int, r : Int;
$\{ M > 0 \}$
 S
$\{ r = \langle \text{Max } i : 0 \leq i \leq M : sum.i - i! \rangle\}$
donde $sum.k = \langle \sum j : 0 \leq j < k : A.j \rangle$

Propuesta de invariante mediante técnica de reemplazo de constante por variable:
$I = r = \langle \text{Max } i : 0 \leq i \leq m : sum.i - i! \rangle \wedge 0 <= m <= M$, de aquí queda definida la guarda $B = m \neq M$.
Comencemos por el cuerpo del ciclo (convenientemente para no repetir la inicialización en caso de tener que reforzar el invariante)
$\{I \wedge B\}$ s1 $\{I\}$

$$wp(r, m := E, m + 1)\ (I)$$
{Definición de wp para asignación}
$$E = \langle \text{Max } i : 0 \leq i \leq m + 1 : sum.i - i! \rangle \wedge 0 <= m + 1 <= M$$
{Suponemos que $0 <= m <= M$ y $m \neq M \rightarrow m + 1 <= M$}
$$E = \langle \text{Max } i : 0 \leq i \leq m + 1 : sum.i - i! \rangle \wedge True$$
{Neutro de la conjunción}
$$E = \langle \text{Max } i : 0 \leq i \leq m + 1 : sum.i - i! \rangle$$
{Aritmética en el rango}
$$E = \langle \text{Max } i : 0 \leq i \leq m \vee i = m + 1 : sum.i - i! \rangle$$
{Partición de rango}
$$E = \langle \text{Max } i : 0 \leq i \leq m : sum.i - i! \rangle \max \langle \text{Max } i : i = m + 1 : sum.i - i! \rangle$$
{Rango unitario en el segundo término, suposición de r en el primero}
$$E = r \max\ sum.(m + 1) - (m + 1)!$$
{Definición de sum}
$$E = r \max \langle \sum j : 0 \leq j < m + 1 : A.j \rangle - (m + 1)!$$
{Veamos que el término A.m es parte de sum, por ende hay problemas de borde}
$$E = r \max \langle \sum j : 0 \leq j < m \vee j = m : A.j \rangle - (m + 1)!$$
{Partición de rango}
$$E = r \max \langle \sum j : 0 \leq j < m : A.j \rangle + \langle \sum j : j = m : A.j \rangle - (m + 1)!$$
{Rango unitario}
$$E = r \max \langle \sum j : 0 \leq j < m : A.j \rangle + A.m - (m + 1)!$$
{Aquí ya no existe problema de borde, puesto que A.m deja de ser término de la sumatoria, propiedad de factorial}

$$E = r \max\langle \sum j : 0 \le j < m : A.j \rangle + A.m - (m + 1) * m!$$

{Fortalecimiento de invariante con $s = \langle \sum j : 0 \le j < m : A.j \rangle$ y $fac = m!$}

Nuevo invariante fortalecido : $I = r = \langle Max\ i : 0 \le i \le m : sum.i - i! \rangle \wedge 0 <= m <= M \wedge s = \sum j : 0 \le j < m : A.j \rangle \wedge fac = m!$
Ahora si buscamos la inicialización tal que $P \rightarrow I$.

$$wp(r, m, s, fac := E, 0, F, G) (I)$$

{Definición de wp para asignación}

$$E = \langle Max\ i : 0 \le i \le 0 : sum.i - i! \rangle \wedge 0 <= 0 <= M \wedge F = \langle \sum j : 0 \le j < 0 : A.j \rangle \wedge G = 0!$$

{Aritmética en el rango del primer término}

$$E = \langle Max\ i : i = 0 : sum.i - i! \rangle \wedge 0 <= 0 <= M \wedge F = \langle \sum j : 0 \le j < 0 : A.j \rangle \wedge G = 0!$$

{Rango unitario y suposición $M => 0$}

$$E = sum.0 - 0! \wedge True \wedge F = \langle \sum j : 0 \le j < 0 : A.j \rangle \wedge G = 0!$$

{Definición de factorial}

$$E = sum.0 - 1 \wedge True \wedge F = \langle \sum j : 0 \le j < 0 : A.j \rangle \wedge G = 1$$

{Rango vacío en el tercer término}

$$E = sum.0 - 1 \wedge True \wedge F = 0 \wedge G = 1$$

{Definición de sum}

$$E = \langle \sum j : 0 \le j < 0 : A.j \rangle - 1 \wedge True \wedge F = 0 \wedge G = 1$$

{Rango vacío}

$$E = 0 - 1 \wedge True \wedge F = 0 \wedge G = 1$$

{Absorbente de la conjunción}

$$E = - 1 \wedge F = 0 \wedge G = 1$$

{Elijo $E \leftarrow -1$, $F \leftarrow 0$ y $G \leftarrow 1$}

True

Inicialización S0 ;
r, m, s, fac := -1, 0, 0, 1;
Ahora encontremos el cuerpo del ciclo con el nuevo invariante reforzado :

$$wp(r, m, s, fac := E, m + 1, G, F) (I)$$

{Definición de wp}

$$E = \langle Max\ i : 0 \le i \le m + 1 : sum.i - i! \rangle \wedge 0 <= m + 1 <= M \wedge G = \sum j : 0 \le j < m + 1 : A.j \rangle \wedge F = m + 1!$$

{Suposición de $0 <= m <= M$ y $m \neq M$}

$$E = \langle Max\ i : 0 \le i \le m + 1 : sum.i - i! \rangle \wedge True \wedge G = \langle \sum j : 0 \le j < m + 1 : A.j \rangle \wedge F = m + 1!$$

{Absorbente de la conjunción y aritmética en los rangos}

$$E = \langle Max\ i : 0 \le i \le m \vee i = m + 1 : sum.i - i! \rangle \wedge G = \langle \sum j : 0 \le j < m \vee i = m : A.j \rangle \wedge F = m + 1!$$

{Partición de rango en ambas cuantificaciones y rango unitario}

$$E = (\langle Max\ i : 0 \le i \le m : sum.i - i! \rangle \max sum.(m + 1) - (m + 1)!) \wedge (G = \sum j : 0 \le j < m : A.j \rangle + A.m) \wedge F = m + 1!$$

{Suposiciones}

$$E = r \max (sum.(m + 1) - (m + 1)!) \wedge G = s + A.m \wedge F = m + 1!$$

{Definición de factorial)

$$E = r \max (sum.(m + 1) - (m + 1) * m!) \wedge G = s + A.m \wedge F = (m + 1) * m!$$

{Suposición de fac}

$$E = r \max (sum.(m + 1) - (m + 1) * fac) \wedge G = s + A.m \wedge F = (m + 1) * fac$$

{Definición de sum}

$$E = r \max (\langle \sum j : 0 \le j < m + 1 : A.j \rangle - (m + 1) * fac) \wedge G = s + A.m \wedge F = (m + 1) * fac$$

{Partición de rango, suposición y rango unitario}

E = (r max (r + A.m - (m + 1) * fac) ^ G = s + A.m ^ F = (m + 1) * fac

{Elijo E ← r max (r + A.m - (m + 1) * fac, G ← s + A.m y F ← (m + 1) * fac)

True

Finalmente, programa anotado:

Const M : Int;
Var A : array[0, M) of Int; r, m, s, fac : Int;
r, m, s, fac := -1, 0, 0, 1;
{ M > 0 }
do m ≠ M →
    r, m, s, fac := r max (r + A.m - (m + 1)), m + 1, s + A.m, (m + 1) * fac;
od
{ r = 〈 Max i : 0 ≤ i ≤ M : sum.i - i! 〉}

Const N : Int; A : Array [0,N) of Int;
Var r : Bool;
{N => 0}
S
{r = < ∑ i : 0 <= i < N ^ A.i > sum.A.i : A.i >
donde sum.A.i = < ∑ j : 0 <= j < i : A.j >

Es claro que necesitamos un ciclo para este programa, comenzamos proponiendo un invariante mediante técnica de reemplazo de constante por variable.
I : {r = < ∑ i : 0 <= i < n ^ A.i > sum.A.i : A.i > ^ 0 <= n <= N}, luego de aquí deducimos la guarda B = n ≠ N y el requisito I ^ ¬B → Q queda asegurado.
Comencemos derivando el cuerpo del ciclo, para no perder tiempo en la inicialización en caso de tener que reforzar invariante.

- {I ^ B} s1 {I}

wp(r, n := E, n + 1) (I)

{Definición de wp para asignación}

E = < ∑ i : 0 <= i < n + 1 ^ A.i > sum.A.i : A.i > ^ 0 <= n + 1 <= N

{El lado derecho se reemplaza por True, por disposición (n <= N) ^ (n ≠ N) → (n + 1) <= N}

E = < ∑ i : 0 <= i < n + 1 ^ A.i > sum.A.i : A.i > ^ True

{Neutro de la conjunción, y aritmética en el rango}

E = < ∑ i : (0 <= i < n v i = n) ^ A.i > sum.A.i : A.i >

{Partición de rango}

E = < ∑ i : 0 <= i < n ^ A.i > sum.A.i : A.i > + < ∑ i : i = n ^ A.i > sum.A.i : A.i >

{Suposición en el lado izquierdo}

E = r + < ∑ i : i = n ^ A.i > sum.A.i : A.i >

{Rango unitario y condición}

E = (A.n > sum.A.n → A.n + r
¬A.n > sum.A.n → r + 0
)

{sum.A.n no es programable, refuerzo el invariante con s = sum.A.n}

Nuevo I’ = {I ^ s = < ∑ j : 0 <= j < n : A.j >}

Probemos ahora la inicialización:

wp(r, n, s := E, 0, F) (I’)

{Definición de wp para asignación}

E = < ∑ i : 0 <= i < 0 ^ A.i > sum.A.i : A.i > ^ 0 <= 0 <= N ^ s = < ∑ j : 0 <= j < 0 : A.j >

{Aritmética en el rango dos veces, suposición de 0 <= N}

E = < ∑ i : False ^ A.i > sum.A.i : A.i > ^ True ^ s = < ∑ j : False : A.j >

{Aritmética en el primer rango, absorbente de la conjunción y rango vacío}

E = < ∑ i : False : A.i > ^ s = 0

{Rango vacío}

E = 0 ^ s = 0

{Elijo E ← 0 y s ← 0}

True

Listo, ahora si derivamos el cuerpo del ciclo con el nuevo invariante:

wp(r, n, s := E, n + 1, F) (I’)

{Definición de wp para asignación}

E = < ∑ i : 0 <= i < n + 1 ^ A.i > sum.A.i : A.i > ^ 0 <= n + 1 <= N ^ s = < ∑ j : 0 <= j < n + 1 : A.j >

{Suposición,neutro de la conjunción y aritmética en el rango de las dos cuantificaciones}

E = < ∑ i : 0 <= i < n v i = n ^ A.i > sum.A.i : A.i > ^ s = < ∑ j : 0 <= j < n v i = n : A.j >

{Partición de rango dos veces]

E = < ∑ i : 0 <= i < n ^ A.i > sum.A.i : A.i > + < ∑ i : i = n ^ A.i > sum.A.i : A.i > ^
s = < ∑ j : 0 <= j < n : A.j > + < ∑ j : i = n : A.j >

{Suposición en el primer y tercer término, rango unitario en el último}

E = r + < ∑ i : i = n ^ A.i > sum.A.i : A.i > ^ s = s + A.n

{Elijo s ← s + A.n}

E = r + < ∑ i : i = n ^ A.i > sum.A.i : A.i > ^ True

{Neutro de la conjunción, rango unitario y condición}

E = (A.n > sum.A.n → A.n + r
¬A.n > sum.A.n → r + 0
)

{Suposición}

E = (A.n > s → A.n + r
¬A.n > s → r + 0
)

Finalmente, quedaría probar que la cota propuesta t = N - n es positiva y decrece en el cuerpo del ciclo (trivial). Programa Anotado:

```
Const N : Int; A : Array [0,N) of Int;
Var r, n, s : Int;
r, n, s := 0, 0, 0;
{N => 0}
do n ≠ N →
  if A.n > s →
     r, n, s := A.n + r, n + 1, s + A.n;
     ¬ (A.n > s) →
     r, n, s := r, n + 1, s + A.n;
  fi
od
{r = < ∑ i : 0 <= i < N ^ A.i > sum.A.i : A.i >
Const N : Int, A : array[0, N) of Int;
```

Var r : Int;
$\{ P : N \geq 0 \}$
S
$\{ Q : r = < N\ p, q : 0 \leq p < q < N : (A.p + A.q) \bmod 2 \neq 0 > \}$

Comencemos la derivación proponiendo el invariante I mediante la técnica de reemplazo de constante por variable:
$I = r = < N\ p, q : 0 \leq p < q < n : (A.p + A.q) \bmod 2 \neq 0 > \wedge\ 0 <= n <= N$
Luego propongo la guarda $B = (N \neq n)$
Derivemos el cuerpo del ciclo S1, a fin de prevenir una inicialización fallida en el caso de necesitar un refuerzo de invariante.

$$wp(r, n := E, n + 1)\ (I)$$
{Definición de wp para asignación}
$$< N\ p, q : 0 \leq p < q < n + 1 : (A.p + A.q) \bmod 2 \neq 0 > \wedge\ 0 <= n + 1 <= N$$
{Suposición en el lado derecho, $(n <= N \wedge n \neq N) \rightarrow n + 1 <= N$}
$$< N\ p, q : 0 \leq p < q < n + 1 : (A.p + A.q) \bmod 2 \neq 0 > \wedge\ True$$
{Neutro de la conjunción, aritmética en el rango}
$$< N\ p, q : 0 \leq p < q \wedge (q = n \vee q < n) : (A.p + A.q) \bmod 2 \neq 0 >$$
{Distributividad}
$$< N\ p, q : 0 \leq p < q \wedge (q = n) : (A.p + A.q) \bmod 2 \neq 0 > +$$
$$< N\ p, q : 0 \leq p < q \wedge (q < n) : (A.p + A.q) \bmod 2 \neq 0 >$$
{Eliminación de variable en el primer término, aritmética en el segundo}
$$< N\ p : 0 \leq p < n : (A.p + A.n) \bmod 2 \neq 0 > +$$
$$< N\ p, q : 0 \leq p < q < n : (A.p + A.q) \bmod 2 \neq 0 >$$
{Suposición en el segundo término}
$$< N\ p: 0 \leq p < n : (A.p + A.n) \bmod 2 \neq 0 > + r$$
{Tenemos problemas de bordes, veamos que el rango no llega a ser igual a n, pero en el término se indexa en n}

Intentamos quitarnos de encima ese A.n haciendo un análisis por casos, basándonos en la siguiente propiedad : “Recuerde que a + b es impar si y sólo si uno de los números es par y el otro es impar”. Luego :

$$< N\ p : 0 \leq p < n : (A.p + A.n) \bmod 2 \neq 0 > + r$$
{Propiedad suma impar}
$$< N\ p : 0 \leq p < n : (A.p \bmod 2 \neq 0) \vee (A.n \bmod 2 \neq 0) > + r$$
{Análisis por casos}

Caso $(A.n \bmod 2 \neq 0)$:

$$< N\ p : 0 \leq p < n : (A.p \bmod 2 \neq 0) \vee True > + r$$
{Absorbente de la disyunción}
$$< N\ p, q : 0 \leq p < n : True > + r$$
{Estamos contando True n veces (n veces 1)}
$$r + n$$

Caso ¬ (A.n mod 2 ≠ 0):

< N p : 0 ≤ p < n : (A.p mod 2 ≠ 0) v False > + r

{Elemento neutro de la disyunción}

< N p : 0 ≤ p < n : (A.p mod 2 ≠ 0) > + r

{Acá hacemos fortalecimiento de invariante}

Nuevo invariante I’ = { I ^ B ^ s = < N p : 0 ≤ p < n : (A.p mod 2 ≠ 0) > }. Derivemos el cuerpo del ciclo nuevamente :

wp(r, n, s := E, n + 1, F) (I’)

{Definición de wp}

E = < N p, q : 0 ≤ p < q < n + 1 : (A.p + A.q) mod 2 ≠ 0 > ^ 0 <= n <= N ^

F = < N p : 0 ≤ p < n + 1 : (A.p mod 2 ≠ 0) >

{Suposición con (n <= N ^ n ≠ N) → n <= N}

E = < N p, q : 0 ≤ p < q < n + 1 : (A.p + A.q) mod 2 ≠ 0 > ^ True ^ F = < N p : 0 ≤ p < n + 1 : (A.p mod 2 ≠ 0) >

{Neutro de la conjunción, aritmética en ambos rangos}

E = < N p, q : 0 ≤ p < q ^ (q = n v q < n) : (A.p + A.q) mod 2 ≠ 0 > ^

F = < N p : (p = n) v (0 ≤ p < n) : (A.p mod 2 ≠ 0) >

{Distributividad en el primer término}

E = < N p, q : (0 ≤ p < q ^ q = n) v (0 ≤ p < q ^ q < n) : (A.p + A.q) mod 2 ≠ 0 > ^

F = < N p : (p = n) v (0 ≤ p < n) : (A.p mod 2 ≠ 0) >

{Partición de rango en ambos términos}

E = < N p, q : 0 ≤ p < q ^ q = n : (A.p + A.q) mod 2 ≠ 0 > +

< N p, q : 0 ≤ p < q ^ q < n : (A.p + A.q) mod 2 ≠ 0 > ^

F = < N p : (p = n) : A.p mod 2 ≠ 0 > + < N p : 0 ≤ p < n : (A.p mod 2 ≠ 0) >

{Eliminación de variable en el primer término y aritmética en el rango del segundo}

E = < N p : 0 ≤ p < n : (A.p + A.n) mod 2 ≠ 0 > +

< N p, q : 0 ≤ p < q < n : (A.p + A.q) mod 2 ≠ 0 > ^

F = < N p : (p = n) : A.p mod 2 ≠ 0 > + < N p : 0 ≤ p < n : (A.p mod 2 ≠ 0) >

{Suposición sobre r, y sobre s}

E = < N p : 0 ≤ p < n : (A.p + A.n) mod 2 ≠ 0 > + r ^ s = < N p : (p = n) : A.p mod 2 ≠ 0 > + s

{Propiedad de suma impar en el primer término}

E = < N p : 0 ≤ p < n : (A.p mod 2 ≠ 0) v (A.n mod 2 ≠ 0) > + r ^ F = < N p : (p = n) : A.p mod 2 ≠ 0 > + s

{Rango unitario en el segundo cuantificador}

E = < N p : 0 ≤ p < n : (A.p mod 2 ≠ 0) v (A.n mod 2 ≠ 0) > + r ^

F = (A.n mod 2 ≠ 0 → 1 + s
¬ A.n mod 2 ≠ 0 → s
)

Hasta aquí tenemos un if para determinar la asignación de s, luego hacemos análisis por casos en el otro cuantificador y vemos la asignación de r, finalmente juntamos ambas guardas para que nos quede un solo if. (son las mismas guardas)

Caso (A.n mod 2 ≠ 0):

E = < N p : 0 ≤ p < n : (A.p mod 2 ≠ 0) v True >
{Absorbente de la disyunción}
…
r + n

Caso ¬ (A.n mod 2 ≠ 0):

E = < N p : 0 ≤ p < n : (A.p mod 2 ≠ 0) v False >
{Neutro de la disyunción}
…
r + s

Hasta acá, cuerpo del ciclo:

do n ≠ N →
  if (A.n mod 2 ≠ 0) →
    r, s, n := r + n, 1 + s, n + 1;
  ¬ (A.n mod 2 ≠ 0) →
    r, s, n := r + s, s, n + 1;
  fi
od

Ahora derivemos la inicialización del ciclo:

wp(r, s, n := E, F, 0) (I’)
{Definición de wp para asignación}
E = < N p, q : 0 ≤ p < q < 0 : (A.p + A.q) mod 2 ≠ 0 > ^ 0 <= 0 <= N ^
F = < N p : 0 ≤ p < n 0 : (A.p mod 2 ≠ 0) >
{Suposición N => 0, rango vacío dos veces}
E = 0 ^ True ^ F = 0
{Elijo E ← 0, F ← 0, neutro de la conjunción}
True

Función de cota t = N - n, (pruebas triviales ya realizadas).
Programa final, anotado :
Const N : Int, A : array[0, N) of Int;
Var r, n, s : Int;
r, n, s := 0, 0, 0;
{ P : N ≥ 0 }
do n ≠ N →
  if (A.n mod 2 ≠ 0) →
    r, s, n := r + n, 1 + s, n + 1;

```
    ¬ ( A.n mod 2 ≠ 0 ) →
      r, s, n := r + s, s, n + 1;
    fi
od
{ Q : r = < N p, q : 0 ≤ p < q < N : (A.p + A.q) mod 2 ≠ 0 > }
```

```
Const M : Int;
Var A : array [0, M) of Int, n : Int;
{ M ≥ 0 }
S
{ n = <N i : 0 ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i > }
```

Proponemos el invariante : I = { n = <N i : r ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i > ^ 0 ≤ r ≤ M } y derivemos el cuerpo del ciclo para ahorrarnos una posible inicialización errónea en caso de tener que fortalecer el invariante, la guarda B = (r ≠ 0).

wp(n, r := E, r - 1) (I)

{Definición de wp para asignación}

E = < N i : r - 1 ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i > ^ 0 ≤ r - 1 ≤ M

{Por suposición el lado derecho es correcto}

E = < N i : r - 1 ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i > ^ True

{Neutro de la conjunción, aritmética en el rango}

E = < N i : (i = r - 1 v r ≤ i ≤ M) : < ∑ j : i ≤ j < M : A.j > < i >

{Partición de rango}

E = <N i : i = r - 1 : < ∑ j : i ≤ j < M : A.j > < i > + <N i : r ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i >

{Suposición de r}

E = <N i : i = r - 1 : < ∑ j : i ≤ j < M : A.j > < i > + r

{Rango unitario}

E = < ∑ j : r - 1 ≤ j < M : A.j > < r - 1 + r

{Aritmética en el rango}

E = < ∑ j : j = r - 1 v r ≤ j < M : A.j > < r - 1 + r

{Partición de rango}

E = < ∑ j : r ≤ j < M : A.j > < r - 1 +< ∑ j : j = r - 1 : A.j > < r - 1> + r

{Rango unitario}

E = ((< ∑ j : r ≤ j < M : A.j >) < r - 1) + (A.(r - 1) < r - 1) + r

{Fortalecimiento de invariante con s = < ∑ j : r ≤ j < M : A.j >}

Volvemos a derivar el cuerpo del ciclo con mi invariante fortalecido :
I’ = { n = <N i : r ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i > ^ 0 ≤ r ≤ M ^ s = < ∑ j : r ≤ j < M : A.j > }

wp(n, r, s := E, r - 1, G) (I’)

{Definición de wp para asignación}

E = <N i : r - 1 ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i > ^ 0 ≤ r ≤ M ^ G = < ∑ j : r - 1 ≤ j < M : A.j >

{Suposición y neutro de la conjunción}

E = <N i : r - 1 ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i > ^ G = < ∑ j : r - 1 ≤ j < M : A.j >

{Aritmética en ambos rangos}

E = <N i : i = r - 1 v r ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i > ^ G = < ∑ j : j = r - 1 v r ≤ j < M : A.j >

{Partición de rango en ambos términos}

$$E = <N\ i : r \le i \le M : < \sum j : i \le j < M : A.j > < i > + < \sum j : j = r - 1 : A.j > < r - 1 \wedge$$
$$G = < \sum j : j = r - 1 : A.j > + < \sum j : r \le j < M : A.j >$$
{Suposición de r y de s }
$$E = r + < \sum j : j = r - 1 : A.j > < r - 1 \wedge$$
$$G = < \sum j : j = r - 1 : A.j > + s$$
{Rango unitario en ambos términos}
$$E = r + (A.(r - 1) < r - 1) \wedge G = (A.(r - 1) + s)$$
{Elijo $E \leftarrow r + (A.(r - 1) < r - 1)$, $G \leftarrow (A.(r - 1) + s)$}
True

Derivemos ahora la inicialización:

$$wp(n, r, s := E,F,G)\ (I')$$
{Definición de wp para asignación}
$$E = <N\ i : F \le i \le M : < \sum j : i \le j < M : A.j > < i > \wedge 0 \le F \le M \wedge G = < \sum j : F \le j < M : A.j >$$
{Propongo $F \leftarrow M$}
$$E = <N\ i : M \le i \le M : < \sum j : i \le j < M : A.j > < i > \wedge 0 \le M \le M \wedge G = < \sum j : M \le j < M : A.j >$$
{Suposición y neutro de la conjunción, aritmética en el primer rango}
$$E = <N\ i : i = M : < \sum j : i \le j < M : A.j > < i > \wedge G = < \sum j : M \le j < M : A.j >$$
{Rango unitario, aritmética en el rango de G}
$$E = < \sum j : M \le j < M : A.j > < M \wedge G = < \sum j : False : A.j >$$
{Aritmética en el rango, rango vacío}
$$E = < \sum j : False : A.j > < i \wedge G = 0$$
{Rango vacío, elijo $G \leftarrow 0$}
$$E = 0 < M \wedge G = 0$$
{Elijo $G \leftarrow 0$, $E \leftarrow 0$}
True

Ahora encontremos la función de cota t. Tal que cumpla los requisitos :

- $I \wedge B \rightarrow t => 0$
- $\{I \wedge B \wedge t = T\}\ s\ \{ t < T\}$

$$t => 0$$
{ Suponemos $I \wedge B$ y propuesta de t }
$$M + r => 0$$
{Suposición $(0 \le r \le M) \wedge (r \ne 0) \rightarrow (r > 0) \rightarrow$suma siempre positiva }
True

$$\{I \wedge B \wedge t = T\}\ s\ \{ t < T\}$$
{Propuesta de t}
$$M + r < T$$
{Especificación}
$$M + (r - 1) < T$$
{Burocracia}
$$M + (r - 1) < M + r$$

$$\{Aritmética\}$$
$$M + r - 1 < M + r$$
$$\{Aritmética\}$$
$$-1 < 0$$
$$True$$

Programa final, anotado :

```
Const M : Int;
Var A : array [0, M) of Int,
Var n, r, s : Int;
n, r, s := 0, M, 0;
{ M ≥ 0 }
do r ≠ 0 →
    n, r, s := r + ( A.(r - 1) < r - 1 ), r - 1, A.(r - 1) + s;
od
{ n = <N i : 0 ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i > }
```

```
Const N : Int; A : Array[0, N) of Int;
 Var r : Bool;
{ N ⩾ 0 }
S
{ r = < Σ i : 0 ⩽ i < N ⋀ prod.A.i < A.i : A.i > }
```

Propuesta de invariante mediante técnica de reemplazo de constante por variable:
$I = \{ \langle \Sigma\, i : 0 \leqslant i < n \wedge prod.A.i < A.i : A.i \rangle \wedge 0 \leq n \leq N \}$ y guarda $B = N \neq n$
Comencemos derivando el cuerpo del bucle para evitar una inicialización errónea en caso de tener que reforzar invariante:

$$wp(r, n := E, n + 1)\ (I)$$
$$\{Definición\ de\ wp\}$$
$$E = \langle \Sigma\, i : 0 \leqslant i < n + 1 \wedge prod.A.i < A.i : A.i \rangle \wedge 0 \leq n + 1 \leq N$$
$$\{Aritmética\ en\ el\ rango,\ suposición\ (n \leq N \wedge n \neq N) \rightarrow n + 1 \leq N\}$$
$$E = \langle \Sigma\, i : (0 \leqslant i < n \vee i = n) \wedge prod.A.i < A.i : A.i \rangle \wedge True$$
$$\{Neutro\ de\ la\ conjunción,\ distributividad\ y\ partición\ de\ rango\}$$
$$E = \langle \Sigma\, i : 0 \leqslant i < n \wedge prod.A.i < A.i : A.i \rangle + \langle \Sigma\, i : (i = n \wedge prod.A.i < A.i) : A.i \rangle$$
$$\{Suposición\ de\ r,\ definición\ de\ prod.A.i\}$$
$$E = r + \langle \Sigma\, i : (i = n \wedge \langle \Pi\, j : 0 \leqslant j < i : A.j \rangle < A.i) : A.i \rangle$$
$$\{Rango\ unitario\ y\ condición\}$$
$$E = ((\langle \Pi\, j : 0 \leqslant j < n : A.j \rangle) < A.n \rightarrow r + A.n$$
$$\neg (\langle \Pi\, j : 0 \leqslant j < n : A.j \rangle) < A.n \rightarrow r + 0$$
$$)$$

Veamos que el cuerpo del ciclo es un análisis por casos, y prod.A.i no es programable, propongo entonces un refuerzo de invariante con la introducción de $s = < \Pi\, j : 0 \leqslant j < n : A.j >$.

$I' = \{ < \Sigma\, i : 0 \leqslant i < n \wedge prod.A.i < A.i : A.i > \wedge\ 0 \leq n \leq N \wedge s = < \Pi\, j : 0 \leqslant j < n : A.j > \}$

Derivemos de nuevo el cuerpo del ciclo :

$$wp(r, n, s := E, n + 1, G)\ (I')$$
$$\{\text{Definición de wp}\}$$
$$E = < \Sigma\, i : 0 \leqslant i < n + 1 \wedge prod.A.i < A.i : A.i > \wedge\ 0 \leq n + 1 \leq N\ \wedge$$
$$G = < \Pi\, j : 0 \leqslant j < n + 1 : A.j >$$
$$\{\text{Suposición } (n \leq N \wedge n \neq N) \rightarrow n + 1 \leq N\}$$
$$E = < \Sigma\, i : 0 \leqslant i < n + 1 \wedge prod.A.i < A.i : A.i > \wedge\ True\ \wedge$$
$$G = < \Pi\, j : 0 \leqslant j < n + 1 : A.j >$$
$$\{\text{Neutro de la conjunción, partición de rango}\}$$
$$E = < \Sigma\, i : 0 \leqslant i < n \wedge prod.A.i < A.i : A.i > + < \Sigma\, i : i = n \wedge prod.A.i < A.i : A.i > \wedge$$
$$G = < \Pi\, j : 0 \leqslant j < n : A.j > * < \Pi\, j : j = n : A.j >$$
$$\{\text{Suposición de r y de s, rango unitario}\}$$
$$E = r + < \Sigma\, i : i = n \wedge prod.A.i < A.i : A.i > \wedge$$
$$G = s * A.n$$
$$\{\text{Rango unitario y condición, elijo } G \leftarrow s * A.n\}$$
$$(< \Pi\, j : 0 \leqslant j < n : A.j >) < A.n \rightarrow r + A.n$$
$$\neg\, (< \Pi\, j : 0 \leqslant j < n : A.j >) < A.n \rightarrow r + 0$$
$$)$$
$$\{\text{Suposición de s}\}$$
$$(\ s < A.n \rightarrow r + A.n$$
$$\neg\, s < A.n \rightarrow r + 0$$
$$)$$

Hasta ahora, cuerpo del ciclo:

do $n \neq N \rightarrow$
  if $(s < A.n) \rightarrow$
    $r, n, s := r + A.n, n + 1, s * A.n;$
  $\neg\, (s < A.n) \rightarrow$
    $r, n, s := r, n + 1, s * A.n;$
  fi
od

Encontremos la inicialización para el programa.

$$wp(r, n, s := E, 0, G)\ (I')$$
$$\{\text{Definición de wp}\}$$
$$E = < \Sigma\, i : 0 \leqslant i < 0 \wedge prod.A.i < A.i : A.i > \wedge\ 0 \leq 0 \leq N\ \wedge$$
$$G = < \Pi\, j : 0 \leqslant j < 0 : A.j >$$
$$\{\text{Suposición, aritmética en ambos rangos}\}$$
$$E = < \Sigma\, i : False \wedge prod.A.i < A.i : A.i > \wedge\ True\ \wedge$$
$$G = < \Pi\, j : False : A.j >$$
$$\{\text{Neutro de la conjunción, rango vacío}\}$$

$$E = 0 \wedge G = 1$$
{Elijo $E \leftarrow 0$, $G \leftarrow 1$}
True

Función de cota t = N - n. Siempre positiva y decreciente durante el cuerpo del ciclo.

Programa final, anotado:

```
Const N : Int; A : Array[0, N) of Int;
Var r, n, s : Int;
r, n, s := 0, 0, 1;
{ N ⩾ 0 }
do  n ≠ N →
      if (s < A.n) →
         r, n, s := r + A.n, n + 1, s * A.n;
      ¬ (s < A.n) →
         r, n, s := r, n + 1, s * A.n;
      fi
od
{ r = < Σ i : 0 ⩽ i < N ∧ prod.A.i < A.i : A.i > }
```

```
Const M : Int;
Var a : array [0, M) of Int; r : Bool;
{M ≥ 0}
S
{r = < ∀ i : 0 ≤ i ≤ M : < Σ j : 0 ≤ j < i : a.j i ≤ hN j : 0 ≤ j < i : a.j ≥ 0 > >}Const M : Int;
Var a : array [0, M) of Int; r : Bool;
{M ≥ 0}
S
{r = < ∀ i : 0 ≤ i ≤ M : < Σ j : 0 ≤ j < i : a.j i ≤ hN j : 0 ≤ j < i : a.j ≥ 0 > >}
```

Planteamos invariante mediante la técnica de reemplazo de constante por variable :

r = < ∀ i : 0 ≤ i ≤ m : < Σ j : 0 ≤ j < i : a.j i ≤ < N j : 0 ≤ j < i : a.j ≥ 0 > > ^ 0 ≤ m ≤ M

Refinemos el ciclo y veamos que es de la forma

```
S0;
do  m ≠ M →
      S1;
od
```

Derivemos el primer cuerpo del ciclo, para evitar una posible inicialización errónea en caso de tener que reforzar el invariante.

wp(r, m := E, m + 1) (I)
{Definición de wp}
E = < ∀ i : 0 ≤ i ≤ m + 1 : < Σ j : 0 ≤ j < i : a.j > ≤ < N j : 0 ≤ j < i : a.j ≥ 0 > > ^
0 ≤ m + 1 ≤ M
{Suposición sobre (m ≠ M) ^ (m ≤ M) → (m < M) → (m + 1 ≤ M)}

$E = \langle \forall\, i : 0 \le i \le m + 1 : \langle \Sigma\, j : 0 \le j < i : a.j \rangle \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle \wedge True$
{Elemento neutro de la conjunción, aritmética en el rango}
$E = \langle \forall\, i : 0 \le i \le m \vee i = m + 1 : \langle \Sigma\, j : 0 \le j < i : a.j \rangle \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle$
{Partición de rango}
$E = \langle \forall\, i : 0 \le i \le m : \langle \Sigma\, j : 0 \le j < i : a.j \rangle \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle \wedge$
$\langle \forall\, i : i = m + 1 : \langle \Sigma\, j : 0 \le j < i : a.j \rangle \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle$
{Suposición sobre r}
$E = r \wedge \langle \forall\, i : i = m + 1 : \langle \Sigma\, j : 0 \le j < i : a.j \rangle \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle$
{Rango unitario}
$E = r \wedge \langle \Sigma\, j : 0 \le j < m + 1 : a.j \rangle \le \langle N\, j : 0 \le j < m + 1 : a.j \ge 0 \rangle$
{Aritmética en ambos rangos}
$Er = r \wedge \langle \Sigma\, j : 0 \le j < m \vee j = m : a.j \rangle \le \langle N\, j : 0 \le j < m \vee j = m : a.j \ge 0 \rangle$
{Partición de rango en ambos términos}
$E = r \wedge \langle \Sigma\, j : 0 \le j < m : a.j \rangle + \langle \Sigma\, j : j = m : a.j \rangle \le \langle N\, j : 0 \le j < m \vee j = m : a.j \ge 0 \rangle$
$+ \langle N\, j : j = m : a.j \ge 0 \rangle$
{Rango unitario en el segundo término}
$E = r \wedge (\langle \Sigma\, j : 0 \le j < m : a.j \rangle + a.m) \le \langle N\, j : 0 \le j < m : a.j \ge 0 \rangle$
$+ \langle N\, j : j = m : a.j \ge 0 \rangle$
{Rango unitario en el último término}
$((a.m \ge 0) \rightarrow r \wedge (\langle \Sigma\, j : 0 \le j < m : a.j \rangle + a.m) \le \langle N\, j : 0 \le j < m : a.j \ge 0 \rangle + 1$
$\neg(a.m \ge 0) \rightarrow r \wedge (\langle \Sigma\, j : 0 \le j < m : a.j \rangle + a.m) \le \langle N\, j : 0 \le j < m : a.j \ge 0 \rangle + 0$
$)$
{Es claro que debo reforzar el invariante ya que me quedan dos cuantificaciones no programables}

Nuevo invariante I' = { $r = \langle \forall\, i : 0 \le i \le m : \langle \Sigma\, j : 0 \le j < i : a.j\, i \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle \wedge 0 \le m \le M \wedge s = \langle \Sigma\, j : 0 \le j < m : a.j \rangle \wedge u = \langle N\, j : 0 \le j < m : a.j \ge 0 \rangle$ }

Volvamos a derivar el cuerpo del ciclo :

wp(r, m, s, u := E, m + 1, F, G) (I')
{Definición de wp para asignación}
$E = \langle \forall\, i : 0 \le i \le m + 1 : \langle \Sigma\, j : 0 \le j < i : a.j \rangle \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle \wedge 0 \le m + 1 \le M \wedge$
$F = \langle \Sigma\, j : 0 \le j < m + 1 : a.j \rangle \wedge G = \langle N\, j : 0 \le j < m + 1 : a.j \ge 0 \rangle$
{Suposición sobre $(m \ne M) \wedge (m \le M) \rightarrow (m < M) \rightarrow (m + 1 \le M)$ y neutro de ^}
$E = \langle \forall\, i : 0 \le i \le m + 1 : \langle \Sigma\, j : 0 \le j < i : a.j \rangle \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle \wedge$
$F = \langle \Sigma\, j : 0 \le j < m + 1 : a.j \rangle \wedge G = \langle N\, j : 0 \le j < m + 1 : a.j \ge 0 \rangle$
{Aritmética en los rangos, y partición de rango}
$E = \langle \forall\, i : 0 \le i \le m : \langle \Sigma\, j : 0 \le j < i : a.j\, i\rangle \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle \wedge$
$\langle \forall\, i : i = m + 1 : \langle \Sigma\, j : 0 \le j < i : a.j \rangle \le \langle N\, j : 0 \le j < i : a.j \ge 0 \rangle \rangle$
$F = \langle \Sigma\, j : 0 \le j < m : a.j \rangle + \langle \Sigma\, j : j = m : a.j \rangle \wedge G = \langle N\, j : 0 \le j < m : a.j \ge 0 \rangle +$
$\langle N\, j : j = m : a.j \ge 0 \rangle$

{Suposición de r, s, u}
$E = r \wedge \langle \forall\, i : i = m + 1 : \langle \Sigma\, j : 0 \leq j < i : a.j \rangle \leq \langle N\, j : 0 \leq j < i : a.j \geq 0 \rangle \rangle$
$F = s + \langle \Sigma\, j : j = m : a.j \rangle \wedge G = u + \langle N\, j : j = m : a.j \geq 0 \rangle$
{Rango unitario en dos términos}
$E = r \wedge \langle \Sigma\, j : 0 \leq j < m + 1 : a.j \rangle \leq \langle N\, j : 0 \leq j < m + 1 : a.j \geq 0 \rangle$
$F = (s + a.m) \wedge G = u + \langle N\, j : j = m : a.j \geq 0 \rangle$
{Partición de rango}
$E = r \wedge \langle \Sigma\, j : 0 \leq j < m : a.j \rangle + \langle \Sigma\, j : j = m : a.j \rangle \leq \langle N\, j : 0 \leq j < m : a.j \geq 0 \rangle +$
$\langle N\, j : j = m : a.j \geq 0 \rangle \wedge$
$F = (s + a.m) \wedge G = u + \langle N\, j : j = m : a.j \geq 0 \rangle$
{Rango unitario}
$E = r \wedge (\langle \Sigma\, j : 0 \leq j < m : a.j \rangle + a.m) \leq \langle N\, j : 0 \leq j < m : a.j \geq 0 \rangle +$
$\langle N\, j : j = m : a.j \geq 0 \rangle \wedge$
$F = (s + a.m) \wedge G = (u + \langle N\, j : j = m : a.j \geq 0 \rangle)$
{Suposición de s y de u}
$E = r \wedge (s + a.m) \leq u + \langle N\, j : j = m : a.j \geq 0 \rangle \wedge$
$F = (s + a.m) \wedge G = (u + \langle N\, j : j = m : a.j \geq 0 \rangle)$
{Rango unitario en los contadores}
$((a.m \geq 0) \rightarrow E = r \wedge (s + a.m) \leq u + 1 \wedge F = (s + a.m) \wedge G = u + 1$
$\neg(a.m \geq 0) \rightarrow E = r \wedge (s + a.m) \leq u \wedge F = (s + a.m) \wedge G = u$
$)$
{Elijo $E \leftarrow r \wedge (s + a.m)$, $F \leftarrow s + a.m$, $G \leftarrow u + 1/ u$}
True

Programa hasta ahora :

S0;
do $m \neq M \rightarrow$
  if $(a.m \geq 0) \rightarrow$
    r, s, u, m := r ^ (s + a.m), s + a.m, u + 1;
  ¬ $(a.m \geq 0) \rightarrow$
    r, s, u, m := r ^ (s + a.m), s + a.m, u;
  fi
od

Derivemos una inicialización para el cuerpo del ciclo :

wp(r, s, u, m := E, F, G, 0) (I')
{Definición de wp para asignación}
$E = \langle \forall\, i : 0 \leq i \leq 0 : \langle \Sigma\, j : 0 \leq j < i : a.j \rangle \leq \langle N\, j : 0 \leq j < i : a.j \geq 0 \rangle \rangle \wedge 0 \leq 0 \leq M \wedge$
$F = \langle \Sigma\, j : 0 \leq j < 0 : a.j \rangle \wedge G = \langle N\, j : 0 \leq j < 0 : a.j \geq 0 \rangle$
{Aritmética y suposición}
$E = \langle \forall\, i : i = 0 : \langle \Sigma\, j : 0 \leq j < i : a.j \rangle \leq \langle N\, j : 0 \leq j < i : a.j \geq 0 \rangle \rangle \wedge True \wedge$
$F = \langle \Sigma\, j : False : a.j \rangle \wedge G = \langle N\, j : False : a.j \geq 0 \rangle$
{Elemento neutro de la conjunción, rango unitario, y rango vacío dos veces}

$$E = \langle \Sigma j : 0 \leq j < 0 : a.j \rangle \leq \langle N j : 0 \leq j < 0 : a.j \geq 0 \rangle \wedge$$
$$F = 0 \wedge G = 0$$
{Aritmética}
$$E = \langle \Sigma j : False : a.j \rangle \leq \langle N j : False : a.j \geq 0 \rangle \wedge F = 0 \wedge G = 0$$
{Rango vacío dos veces}
$$E = 0 \leq 0 \wedge F = 0 \wedge G = 0$$
{Aritmética}
$$E = True \wedge F = 0 \wedge G = 0$$
{Elijo E ← True, F ← 0, G ←0}
True

Programa hasta ahora:

```
r, s, u, m := True, 0, 0, 0;
do  m ≠ M →
      if (a.m ≥ 0) →
         r, s, u, m := r ^ (s + a.m), s + a.m, u + 1;
      ¬ (a.m ≥ 0) →
         r, s, u, m := r ^ (s + a.m), s + a.m, u;
      fi
od
```

Función de cota t siempre positiva y decreciente en la ejecución del bucle :
t = M - m. Las pruebas son triviales y salen fácil, no las veremos.

Programa final anotado :

```
Const M : Int;
Var a : array [0, M) of Int;
Var r, s, u, m : Bool, Int, Int, Int;
r, s, u, m := True, 0, 0, 0;
{M ≥ 0}
do  m ≠ M →
      if (a.m ≥ 0) →
         r, s, u, m := r ^ (s + a.m), s + a.m, u + 1;
      ¬ (a.m ≥ 0) →
         r, s, u, m := r ^ (s + a.m), s + a.m, u;
      fi
od
{r = < ∀ i : 0 ≤ i ≤ M : < Σ j : 0 ≤ j < i : a.j i ≤ hN j : 0 ≤ j < i : a.j ≥ 0 > >}
```

Const N : Int, A : array[0, N) of Int;
Var r : Int;
{P : $N \geq 0$}
S
{Q : $r = \langle N\ p, q : 0 \leq p < q < N : A.p * A.q = 0 \rangle$}

Comenzamos proponiendo el invariante $I = \{ r = \langle N\ p, q : 0 \leq p < q < n : A.p * A.q = 0 \rangle \wedge 0 \leq n \leq N \}$, mediante la técnica de reemplazo de constante por variable, luego deducimos la guarda $B = n \neq N$. Luego derivamos el cuerpo del ciclo para evitar inicializar erróneamente en caso de tener que fortalecer el invariante.

$$wp(r, n := E, n + 1)\ (I)$$
{Definición de wp para la asignación}
$$r = \langle N\ p, q : 0 \leq p < q < n + 1 : A.p * A.q = 0 \rangle \wedge 0 \leq n + 1 \leq N$$
{Suposición sobre $(n \neq N) \wedge (n \leq N) \rightarrow (n < N) \rightarrow (n + 1 \leq N)$}
$$r = \langle N\ p, q : 0 \leq p < q < n + 1 : A.p * A.q = 0 \rangle \wedge True$$
{Elemento neutro de la conjunción, aritmética en el rango}
$$r = \langle N\ p, q : 0 \leq p < q \wedge (q < n \vee q = n) : A.p * A.q = 0 \rangle$$
{Distributividad y partición de rango}
$$r = \langle N\ p, q : 0 \leq p < q \wedge (q < n) : A.p * A.q = 0 \rangle + \langle N\ p, q : 0 \leq p < q \wedge (q = n) : A.p * A.q = 0 \rangle$$
{Aritmética en el primer rango, eliminación de variable en el segundo}
$$r = \langle N\ p, q : 0 \leq p < q < n : A.p * A.q = 0 \rangle + \langle N\ p, q : 0 \leq p < n : A.p * A.n = 0 \rangle$$
{Suposición sobre r}
$$r = r + \langle N\ p, q : 0 \leq p < n : A.p * A.n = 0 \rangle$$
{Tenemos problemas de bordes, no podemos fortalecer. Proponemos un ciclo anidado}

Hasta ahora el programa :

```
S0;
do (n ≠ N) →
    S1;
    do ( …. )→
        S2;
    od
S3;
od
```

Planteemos el invariante para el ciclo anidado tal que $I' = \{ I \wedge B \wedge c = \langle N\ p, q : 0 \leq p < k : A.p * A.n = 0 \rangle \wedge 0 \leq k \leq n \}$ y derivemos su inicialización.

$$wp\ (c, k := E, 0)\ (I')$$
{Definición de wp para asignación}
$$I \wedge B \wedge E = \langle N\ p, q : 0 \le p < 0 : A.p * A.n = 0 \rangle \wedge 0 \le 0 \le n$$
{Suposición y aritmética en el rango}
$$True \wedge E = \langle N\ p, q : False : A.p * A.n = 0 \rangle \wedge True$$
{Neutro de la conjunción y rango vacío}
$$E = 0$$

Hasta ahora el programa :

```
S0;
do (n ≠ N) →
    c, k := 0, 0;
    do (k ≠ n)→
        S2;
    od
S3;
od
```

Ahora derivemos el cuerpo del ciclo :

$$wp(c, k := E, k + 1)\ (I')$$
{Definición de wp par asignación}
$$I \wedge B \wedge E = \langle N\ p, q : 0 \le p < k + 1 : A.p * A.n = 0 \rangle \wedge 0 \le k + 1 \le n$$
{Suposición $(0 \le k) \to (0 \le k + 1) \to (k \ne n) \to (k < n) \to (k + 1 \le n)$}
$$I \wedge B \wedge E = \langle N\ p, q : 0 \le p < k + 1 : A.p * A.n = 0 \rangle \wedge True$$
{Suposición $I \wedge B$, aritmética en el rango y neutro de la conjunción}
$$E = \langle N\ p, q : 0 \le p < k \vee p = k : A.p * A.n = 0 \rangle$$
{Partición de rango}
$$E = \langle N\ p, q : 0 \le p < k : A.p * A.n = 0 \rangle + \langle N\ p, q : p = k : A.p * A.n = 0 \rangle$$
{Suposición sobre c}
$$E = c + \langle N\ p, q : p = k : A.p * A.n = 0 \rangle$$
{Rango unitario}
$$\begin{array}{l} ((A.k * A.n = 0) \to c + 1 \\ \neg (A.k * A.n = 0) \to c \\ ) \end{array}$$

Programa, hasta ahora :

```
S0;
do (n ≠ N) →
    c, k := 0, 0;
    do (k ≠ n)→
        if (A.k * A.n = 0) →
            c, k := c + 1, k + 1;
            ¬ (A.k * A.n = 0) →
            c, k : = c, k + 1;
        fi
    od
r, n := r + c, n + 1;
od
```

Nos falta encontrar la inicialización del primer ciclo, derivemos entonces :

$$wp(r, n := E, 0)\ (I)$$
$$\{\text{Definición de wp para asignación}\}$$
$$E = \langle N\ p, q : 0 \leq p < q < 0 : A.p * A.q = 0 \rangle \wedge 0 \leq 0 \leq N$$
$$\{\text{Suposición y aritmética en el rango}\}$$
$$E = \langle N\ p, q : False : A.p * A.q = 0 \rangle \wedge True$$
$$\{\text{Elemento neutro de la conjunción, rango vacío}\}$$
$$E = 0$$

Programa final, anotado :

```
Const N : Int, A : array[0, N) of Int;
Var r, n, c, k : Int;
r, n := 0, 0;
{P : N ≥ 0}
do (n ≠ N) →
    c, k := 0, 0; c, k := 0, 0;
    do (k ≠ n)→
        if (A.k * A.n = 0) →
            c, k := c + 1, k + 1;
            ¬ (A.k * A.n = 0) →
            c, k : = c, k + 1;
        fi
    od
r, n := r + c, n + 1;
od
{Q : r =  < N p, q : 0 ≤ p < q < N : A.p ∗ A.q = 0 >}
```

Const N : Int, A : array [0, N) of Int;
Var r : Bool;
{P : N ≥ 0}
S
{Q : r = < ∃ i : 0 ≤ i ≤ N : < ∑ j : 0 ≤ j < i : A.j > = -1 >

Primero proponemos el invariante I mediante la técnica de reemplazo de constante por variable, de esta forma el requisito (I ^ ¬ B) → Q queda probado. Luego el invariante es I = { r = < ∃ i : 0 ≤ i ≤ n : < ∑ j : 0 ≤ j < i : A.j > = -1 > ^ 0 ≤ n ≤ N } y además la guarda B = n ≠ N se puede deducir. Derivemos entonces el cuerpo del ciclo para evitar no forzar una inicialización errónea en caso de tener que fortalecer.

wp(r, n := E, n + 1) (I)
{Definición de wp para asignación}
E = < ∃ i : 0 ≤ i ≤ n + 1 : < ∑ j : 0 ≤ j < i : A.j > = -1 > ^ 0 ≤ n + 1 ≤ N
{Suposición sobre (n ≠ N) ^ (n ≤ N) → (n < N) → (n + 1 ≤ N)}
E = < ∃ i : 0 ≤ i ≤ n + 1 : < ∑ j : 0 ≤ j < i : A.j > = -1 > ^ True
{Elemento neutro de la conjunción, aritmética en el rango}
E = < ∃ i : 0 ≤ i ≤ n v i = n + 1 : < ∑ j : 0 ≤ j < i : A.j > = -1 >
{Partición de rango}
E = < ∃ i : 0 ≤ i ≤ n : < ∑ j : 0 ≤ j < i : A.j > = -1 > v
< ∃ i : i = n + 1 : < ∑ j : 0 ≤ j < i : A.j > = -1 >
{Suposición sobre r, rango unitario}
E = r v < ∑ j : 0 ≤ j < n + 1 : A.j > = -1
{Hay problema de borde, trabajo el rango para poder fortalecer}
E = r v < ∑ j : 0 ≤ j < n v j = n : A.j > = -1
{Partición de rango}
E = r v < ∑ j : 0 ≤ j < n : A.j > + < ∑ j : j = n : A.j > = -1
{Rango unitario}
E = r v (< ∑ j : 0 ≤ j < n : A.j > A.n = -1)
{Fortalecimiento de invariante}

I’ = { r = < ∃ i : 0 ≤ i ≤ n : < ∑ j : 0 ≤ j < i : A.j > = -1 > ^ s = < ∑ j : 0 ≤ j < n : A.j > ^ 0 ≤ n ≤ N }

Derivemos de nuevo el cuerpo del ciclo con el nuevo invariante :

wp(r, s, n := E, G, n + 1) (I’)
{Definición de wp para asignación}
E = < ∃ i : 0 ≤ i ≤ n + 1 : < ∑ j : 0 ≤ j < i : A.j > = -1 > ^ G = < ∑ j : 0 ≤ j < n + 1 : A.j > ^ 0 ≤ n + 1 ≤ N
{Suposición sobre (n ≠ N) ^ (n ≤ N) → (n < N) → (n + 1 ≤ N)}
E = < ∃ i : 0 ≤ i ≤ n + 1 : < ∑ j : 0 ≤ j < i : A.j > = -1 > ^ G = < ∑ j : 0 ≤ j < n + 1 : A.j > ^ True

{Elemento neutro de la conjunción, aritmética en los rangos}

$$E = \langle \exists\, i : 0 \le i \le n \vee i = n + 1 : \langle \sum j : 0 \le j < i : A.j \rangle = -1 \rangle \wedge$$
$$G = \langle \sum j : 0 \le j < n \vee i = n : A.j \rangle$$

{Partición de rango en ambos términos}

$$E = \langle \exists\, i : 0 \le i \le n : \langle \sum j : 0 \le j < i : A.j \rangle = -1 \rangle \vee$$
$$\langle \exists\, i : i = n + 1 : \langle \sum j : 0 \le j < i : A.j \rangle = -1 \rangle \wedge$$
$$G = \langle \sum j : 0 \le j < n : A.j \rangle + \langle \sum j : j = n : A.j \rangle$$

{Rango unitario en dos términos}

$$E = \langle \exists\, i : 0 \le i \le n : \langle \sum j : 0 \le j < i : A.j \rangle = -1 \rangle \vee \langle \sum j : 0 \le j < n + 1 : A.j \rangle = -1 \wedge$$
$$(G = \langle \sum j : 0 \le j < n : A.j \rangle + A.n)$$

{Suposición sobre r, s}

$$E = r \vee \langle \sum j : 0 \le j < n + 1 : A.j \rangle = -1 \wedge (G = s + A.n)$$

{Aritmética en el rango}

$$E = r \vee \langle \sum j : 0 \le j < n \vee j = n : A.j \rangle = -1 \wedge (G = s + A.n)$$

{Partición de rango}

$$E = r \vee \langle \sum j : 0 \le j < n : A.j \rangle + \langle \sum j : j = n : A.j \rangle = -1 \wedge (G = s + A.n)$$

{Rango unitario}

$$E = r \vee (\langle \sum j : 0 \le j < n : A.j \rangle + A.n = -1) \wedge (G = s + A.n)$$

{Suposición sobre s}

$$E = r \vee (s + A.n = -1) \wedge (G = s + A.n)$$

{Elijo $E \leftarrow r \vee (s + A.n = -1)$ y $G \leftarrow s + A.n$}

True

Bien, ahora encontremos la inicialización para estas variables :

$$wp(r, s, n := E, G, 0)\ (I')$$

{Definición de wp para asignación}

$$E = \langle \exists\, i : 0 \le i \le 0 : \langle \sum j : 0 \le j < i : A.j \rangle = -1 \rangle \wedge G = \langle \sum j : 0 \le j < 0 : A.j \rangle \wedge$$
$$0 \le 0 \le N$$

{Suposición y neutro de la conjunción ,aritmética en el primer rango}

$$E = \langle \exists\, i : i = 0 : \langle \sum j : 0 \le j < i : A.j \rangle = -1 \rangle \wedge G = \langle \sum j : 0 \le j < 0 : A.j \rangle$$

{Rango unitario, aritmética en el rango de G]

$$E = \langle \sum j : 0 \le j < 0 : A.j \rangle = -1 \wedge G = \langle \sum j : False : A.j \rangle$$

{Aritmética en el rango, rango vacío}

$$E = \langle \sum j : False : A.j \rangle = -1 \wedge G = 0$$

{Rango vacío}

$$E = (0 = -1) \wedge G = 0$$

{Aritmética}

$$E = False \wedge G = 0$$

{Elijo $E \leftarrow False$ y $G \leftarrow 0$}

True

Función de cota t siempre positiva y decreciente en las ejecuciones del ciclo. Propuesta t = N - n (pruebas fáciles y pasos triviales ya resueltos anteriormente).

Programa final, anotado :

Const N : Int, A : array [0, N) of Int;
Var r : Bool;
Var s, n : Int;
r, n, s := False, 0, 0;
{P : N ≥ 0}
do n ≠ N →
    r, n, s := r v (s + A.n = -1), n + 1, s + A.n
od
{Q : r = < ∃ i : 0 ≤ i ≤ N : < ∑ j : 0 ≤ j < i : A.j > = -1 >

Const N : Int, A : array[0, N) of Int;
Var r : Int;
{P : N ≥ 0}
S
{Q : r = < ∏ i, j : 0 ≤ i < j < N : A.i + A.j >}

Comenzamos proponiendo el invariante mediante técnica de reemplazo de constante por variable, tal que I = { r = < ∏ i, j : 0 ≤ i < j < n : A.i + A.j > ^ 0 ≤ n ≤ N}. Y derivemos el cuerpo del ciclo, teniendo en cuenta que la guarda es B = n ≠ N.

wp(n, r := n + 1, E) (I)
{Definición de wp}
E = < ∏ i, j : 0 ≤ i < j < n + 1 : A.i + A.j > ^ 0 ≤ n + 1 ≤ N
{Suposición sobre (n ≠ N) ^ (n ≤ N) → (n < N) → (n + 1 ≤ N)}
E = < ∏ i, j : 0 ≤ i < j < n + 1 : A.i + A.j > ^ True
{Elemento neutro de la conjunción}
E = < ∏ i, j : 0 ≤ i < j < n + 1 : A.i + A.j >
{Aritmética en el rango}
E = < ∏ i, j : 0 ≤ i < j ^ (j = n v j < n) : A.i + A.j >
{Distributividad}
E = < ∏ i, j : (0 ≤ i < j ^ j = n) v (0 ≤ i < j ^ j < n) : A.i + A.j >
{Partición de rango}
E = < ∏ i, j : (0 ≤ i < j ^ j = n) : A.i + A.j > *
< ∏ i, j : (0 ≤ i < j ^ j < n) : A.i + A.j >
{Eliminación de variable con j = n, aritmética en el rango}
E = < ∏ i : 0 ≤ i < n : A.i + A.n > *
< ∏ i, j : 0 ≤ i < j < n : A.i + A.j >
{Suposición sobre r}
E = < ∏ i : 0 ≤ i < n : A.i + A.n > * r

{Tenemos problemas de bordes, no podemos fortalecer, propongo un ciclo anidado}
E = < ∏ i : 0 ≤ i < n : A.i + A.n > * r
{Introducción s = < ∏ i : 0 ≤ i < n : A.i + A.n >}
E = s * r

Veamos el segundo ciclo, propongo el invariante I' = { I ^ B ^ s = < ∏ i : 0 ≤ i < k : A.i + A.n > ^ 0 ≤ k ≤ n } y derivemos el cuerpo del segundo ciclo con guarda B = k ≠ n.

wp (s, k := E, k + 1) (I')
{Definición de wp para asignación}
I ^ B ^ E = < ∏ i : 0 ≤ i < k + 1 : A.i + A.n > ^ 0 ≤ k + 1 ≤ n
{Suposición sobre I ^ B → True y (0 ≤ k ≤ n) ^ (k ≠ n) → (k < n) →(k + 1 ≤ n)}
True ^ E = < ∏ i : 0 ≤ i < k + 1 : A.i + A.n > ^ True
{Elemento neutro de la conjunción, aritmética en el rango}
E = < ∏ i : 0 ≤ i < k v i = k : A.i + A.n >
{Partición de rango}
E = < ∏ i : 0 ≤ i < k : A.i + A.n > * < ∏ i : i = k : A.i + A.n >
{Suposición sobre s y rango unitario}
E = s * (A.k + A.n)
{Elijo E ← A.k + A.n}
True

Derivemos las inicializaciones de los dos ciclos, comenzando con el primero.

wp(r, n := E, 0) (I)
{Definición de wp para asignación}
E = < ∏ i, j : 0 ≤ i < j < 0 : A.i + A.j > ^ 0 ≤ 0 ≤ N
{Suposición y aritmética en el rango}
E = < ∏ i, j : False : A.i + A.j > ^ True
{Elemento neutro de la conjunción y rango vacío}
E = 1
{Elijo E ← 1}
True

wp (s, k := E, 0) (I')
{Definición de wp para asignación}
I ^ B ^ E = < ∏ i : 0 ≤ i < 0 : A.i + A.n > ^ 0 ≤ 0 ≤ n
{Suposición sobre I ^ B y 0 ≤ n, elemento neutro de la conjunción}
E = < ∏ i : 0 ≤ i < 0 : A.i + A.n >
{Aritmética en el rango y rango vacío}
E = 1
{Elijo E ← 1}
True

Programa hasta ahora :

```
r, n := 1, 0;
do n ≠ N →
    s, k := 1, 0;
    do k ≠ n →
        s, k :=  s * (A.k + A.n), k + 1;
    od
r, n := r * s, n + 1;
```

Funciones de cota t = (N - n) y t’ = (k - n), siempre positivas y decrecientes durante la ejecución del bucle (pruebas triviales y sencillas ya vistas).

Programa final, anotado :

```
Const N : Int, A : array[0, N) of Int;
Var r, n, s, k : Int;
r, n := 1, 0;
{P : N ≥ 0}
do n ≠ N →
    s, k := 1, 0;
    do k ≠ n →
        s, k :=  s * (A.k + A.n), k + 1;
    od
r, n := r * s, n + 1;
od
{Q : r = < ∏ i, j : 0 ≤ i < j < N : A.i + A.j >}
```

Especificaciones y corrida de ejemplos en el paradigma Imperativo

- Dados dos arreglos A y B determinar si todos los elementos de en A son mayores a algún elemento de B.

Const M : Int;
Var r : Bool;
    a : array [0,M) of Int;
    b : array [0,N) of int;
{ M => 0 ^ N => 0 }
S
{ r = < ∀ i : 0 <= i < M : < ∃ j: 0 <= j < N : b.j < a.i > }

- Dado un arreglo A decir si la suma de los elementos de algún segmento del mismo es mayor a cero.

Const N : Int;
Var r : Bool;
    a : array [0,M) of Int;
    b : array [0,N) of int;
{ M ≥ 0 }
S
{ r = < ∃ p, q : 0 ≤ p < q < N : < Σ i : p ≤ i ≤ q : a.i > > 0 > }

- Dado un arreglo de al menos 2 elementos, decidir (calcular) si hay dos elementos consecutivos cuya diferencia sea menor a una constante K.

Const N ,k : Int;
Const a : array [0,N) of Int;
Var r : Bool;
{ N ≥ 1 }
S
{ r = < ∃ p : 0 ≤ p ≤ N - 1 : A.p - A.p+1 < k > }

Const M : Int;
Var A : array [0, M) of Int, n : Int;
{M ≥ 0}
S
{n = < N i : 0 ≤ i ≤ M : < ∑ j : i ≤ j < M : A.j > < i >}

- Calcula el resultado para A = [3, −1, 1, −1] usando la especificación. Justifica, enumerando todos los elementos del rango del contador.

$$n = < N\ i : 0 \leq i \leq M : < \sum j : i \leq j < M : A.j > < i >$$

{Aplicamos la cuantificación al arreglo A = [3, −1, 1, −1]}

$$n = < N\ i : i\ e\ \{0,1,2,3,4\} : < \sum j : i \leq j < M : A.j > < i >$$

{Aplicamos el término a cada posible “i”}

$i = 0 \rightarrow j\ e\ \{0,1,2,3\}$

$$< \sum j : j\ e\ \{0,1,2,3\} : A.j > < 0$$

{Términos aplicados a los posibles valores de j}

$$A.0 + A.1 + A.2 + A.3 < 0$$

{Indexación}

$$3\ -1 + 1 -1 + 0 < 0$$

{Aritmética}

$$2 < 0$$

{Lógica}

False

$i = 1 \rightarrow j\ e\ \{1,2,3\}$

$$< \sum j : j\ e\ \{1,2,3\} : A.j > < 1$$

{Términos aplicados a los posibles valores de j}

$$A.1 + A.2 + A.3 < 0$$

{Indexación}

$$-1 + 1 -1 + 0 < 0$$

{Aritmética}

$$-1 < 0$$

{Lógica}

True

$i = 2 \rightarrow j\ e\ \{2,3,4\}$

$$< \sum j : j\ e\ \{2,3\} : A.j > < 2$$

{Términos aplicados a los posibles valores de j}

$$A.2 + A.3$$

{Indexación}

$$1 - 1 + 0 < 2$$

{Aritmética}

$$0 < 2$$

{Lógica}

True

$i = 3 \rightarrow j\ e\ \{3,4\}$

$$< \sum j : j\ e\ \{3\}: A.j > < 3$$

{Términos aplicados a los posibles valores de j}

$$A.3 < 3$$

$$\{\text{Indexación}\}$$
$$-1 < 3$$
$$\{\text{Lógica}\}$$
$$\text{True}$$

$i = 4 \rightarrow j \in \{4\}$

$$\langle \sum j : \text{False} : A.j \rangle < 0$$
$$\{\text{Rango vacío}\}$$
$$0 < 0$$
$$\text{False}$$

Número de veces el cual el término se hizo True es 3, por lo tanto :

$$n = \langle N\ i : i \in \{0,1,2,3\} : \langle \sum j : i \leq j < M : A.j \rangle < i \rangle$$
$$\ldots$$
$$n = 3$$