

Derivación de Programas Funcionales

Programación Avanzada
UNRC
Pablo Castro

Derivación de Programas

La derivación de programas nos permite obtener programas correctos a partir de su especificación.

Técnicas:

- Inducción,
- Modularización,
- Cambiar constantes por variables,
- Tupling,
- Generalización

Usando Inducción

La técnica de inducción se basa en las siguientes ideas:

- Para los casos bases reemplazamos en la especificación los parámetros con los valores de casos bases.
- Para el caso inductivo, trabajamos con la especificación utilizando como hipótesis que la función a definir funciona correctamente para los valores anteriores al actual.

Cambiar Constantes por Variables

Consideramos que queremos calcular es siguiente número:

$$\langle \sum i : 0 \leq i \leq N : X^i \rangle$$

En vez de calcular el número directamente, podemos considerar la siguiente función:

$$\langle \forall n :: f.n = \langle \sum i : 0 \leq i < n : X^i \rangle \rangle$$

N una constante es cambiada por una variable,

Derivando por Inducción

Vamos a derivar una función utilizando la especificación:

Caso Base:

$$\begin{aligned} f.0 &= [\text{definición}] \\ &\langle \sum i : 0 \leq i < 0 : X^i \rangle \\ &= [\text{Rango vacío}] \\ 0 & \end{aligned}$$

Caso Inductivo:

$$\begin{aligned} f.(n+1) &= [\text{definición}] \\ &\langle \sum i : 0 \leq i < (n+1) : X^i \rangle \\ &= [\text{Aritmetica}] \\ &\langle \sum i : 0 \leq i < n \vee n \leq i < n+1 : X^i \rangle \\ &= [\text{Aritmetica}] \\ &\langle \sum i : 0 \leq i < n \vee i = n : X^i \rangle \\ &= [\text{Partición de rango y rango único}] \\ &\langle \sum i : 0 \leq i < n : N^i \rangle + X^n \\ &= [\text{Inducción}] \\ f.n + X^n & \end{aligned}$$

Modularizando

La función anterior queda: $f : \text{Num} \rightarrow \text{Num}$

$$f.0 \doteq 0$$

$$f.(n + 1) \doteq f.n + X^n$$

Para calcular X^n podemos agregar una función local:

$$f : \text{Num} \rightarrow \text{Num}$$

$$f.0 = 0$$

$$f.(n + 1) = g.n + f.n$$

$$\llbracket g.i = X^i \rrbracket$$

Ejercicio: Derivar g

Otra Derivación

Podemos derivar la función de otra forma:

$$\begin{aligned} f.(n+1) &= [\text{definición}] \\ \langle \sum i : 0 \leq i < (n+1) : X^i \rangle &= [\text{Aritmetica}] \\ \langle \sum i : 0 \leq i < 1 \vee 1 \leq i < n+1 : X^i \rangle &= [\text{Partición de rango y Rango único}] \\ X^0 + \langle \sum i : 1 \leq i < (n+1) : X^i \rangle &= [\text{Reemplazando i por j+1}] \\ X^0 + \langle \sum j : 1 \leq j + 1 < (n+1) : X^{j+1} \rangle &= [\text{Aritmética}] \\ X^0 + \langle \sum j : 1 \leq j + 1 < (n+1) : X^j * X \rangle &= [\text{Prop. Cuantificadores}] \\ X^0 + X * \langle \sum j : 1 \leq j + 1 < (n+1) : X^j \rangle &= [\text{Aritmética}] \\ X^0 + X * \langle \sum j : 0 \leq j < n : X^j \rangle &= [\text{Inducción}] \\ X^0 + X * f.n & \end{aligned}$$

Hace menos
multiplicaciones que
antes

Tupling

Muchas veces podemos usar tuplas para hacer más eficientes los programas:

fib : Num → Num

fib.0 = 0

fib.1 = 1

fib.(n + 2) = fib.(n + 1) + fib.n

Función fibonacci

Esta definición de fibonacci toma tiempo exponencial.

Mejorando la Eficiencia

Podemos calcular la siguiente función:

$$\begin{aligned}g &: \text{Num} \rightarrow (\text{Num}, \text{Num}) \\g.n &= (\text{fib}.n, \text{fib}.(n + 1))\end{aligned}$$

Para el caso base:

$$\begin{aligned}g.0 &= [\text{Def.g}] \\&= (\text{fib}.0, \text{fib}.1) \\&= [\text{Def.fib}] \\&= (0, 1)\end{aligned}$$

Usando Tuplas

Veamos el caso inductivo:

Se recalcula $\text{fib}.\text{(n+1)}$

$$\begin{aligned} g.(n+1) &= \text{Def. } g \\ (\text{fib}.\text{(n+1)}, \text{fib}.\text{(n+2)}) &= \text{Def. fib} \\ (\text{fib}.\text{(n+1)}, \text{fib}.\text{n} + \text{fib}.\text{(n+1)}) &= [\text{intro. de } a \text{ y } b] \\ (\text{fib}.\text{(n+1)}, \text{fib}.\text{n} + \text{fib}.\text{(n+1)}) &\\ \llbracket a = \text{fib}.\text{n} & \\ b = \text{fib}.\text{(n+1)} \rrbracket & \\ = [\text{igualdad de pares}] & \\ (\text{fib}.\text{(n+1)}, \text{fib}.\text{n} + \text{fib}.\text{(n+1)}) & \\ \llbracket (a, b) = (\text{fib}.\text{n}, \text{fib}.\text{(n+1)}) \rrbracket & \\ = [\text{reemplazo}] & \\ (b, a + b) & \\ \llbracket (a, b) = (\text{fib}.\text{n}, \text{fib}.\text{(n+1)}) \rrbracket & \\ = [\text{Inducción}] & \\ (b, a + b) & \\ \llbracket (a, b) = g.\text{n} \rrbracket & \end{aligned}$$

Usando Tuplas

La función queda:

$$f : \text{Num} \rightarrow \text{Num}$$

$$g.0 = (0, 1)$$

$$g.(n + 1) = (b, a + b)$$

$$\llbracket (a, b) = g.n \rrbracket$$

Definimos:

$$fib.n = (g.n).0$$

Esta función es lineal
en cuanto a n

Generalización

Cuando no se puede aplicar la hipótesis inductiva se puede cambiar la especificación:

$$P : [Num] \rightarrow Num$$

$$P.xs = \langle \forall i : 0 \leq i \leq \#xs : sum.(xs \uparrow i) \geq 0 \rangle$$

$$\begin{aligned} [] \uparrow n &= [] \\ xs \uparrow 0 &= [] \\ (x \triangleright xs) \uparrow (n + 1) &= x \triangleright (xs \uparrow n) \end{aligned}$$

Tratar de derivar la especificación para el caso inductivo no es factible:

$$\begin{aligned} P.(x \triangleright xs) & \\ = [def.P] & \\ \langle \forall i : 0 \leq i \leq \#x \triangleright xs : sum.((x \triangleright xs) \uparrow i) \geq 0 \rangle & \\ = [def.\#] & \\ \langle \forall i : 0 \leq i \leq 1 + \#xs : sum.((x \triangleright xs) \uparrow i) \geq 0 \rangle & \\ = [\text{Part. de rango y logica}] & \\ True \wedge \langle \forall i : 0 \leq i \leq \#xs : x + sum.(xs \uparrow i) \geq 0 \rangle & \end{aligned}$$

En este punto no se puede aplicar la hipótesis inductiva

Generalización

Podemos generalizar la **especificación**:

$$Q : Num \rightarrow [Num] \rightarrow Bool$$

$$Q.n.xs = \langle \forall i : 0 \leq i \leq \#xs : n + sum.(xs \uparrow i) \geq 0 \rangle$$

Esta función es más general que la original:

$$P.xs = Q.0.xs$$

Trataremos de derivar esta función.

Generalización

Derivemos Q:

$$\begin{aligned} & Q.(x \triangleright xs) \\ &= [\text{Def. Q}] \\ & \langle \forall i : 0 \leq i \leq \#(x \triangleright xs) : n + (\text{sum}.((x \triangleright xs) \uparrow (i + 1))) \geq 0 \rangle \\ &= [\text{Def. } \# \text{ y Def. } \uparrow] \\ & \langle \forall i : 0 \leq i \leq 1 + \#xs : n + (\text{sum}.(x \triangleright (xs \uparrow i))) \geq 0 \rangle \\ &= [\text{separacion de un termino, y def. sum}] \\ & n + x \geq 0 \wedge \langle \forall i : 0 \leq i \leq \#xs : n + x + (\text{sum}.((xs \uparrow i))) \geq 0 \rangle \\ &= [H.I] \\ & n + x \geq 0 \wedge Q.n + x.xs \end{aligned}$$

Es decir:

$$Q.n.[] = n \geq 0$$

$$Q.n.(x \triangleright xs) = n \geq 0 \wedge Q.(n + x).xs$$

Entonces: $P.xs = Q.0.xs$

Ejemplos con Subsegmentos

Veamos un ejemplo con subsegmentos.

$$f : [Num] \mapsto Num$$

$$f.xs = \langle \text{Min } as, bs, cs : xs = as ++ bs ++ cs : sum.bs \rangle$$

Problema del
subsegmento de
suma minima

El caso base queda como ejercicio, y obtenemos:

$$f.[] = 0$$

Problemas con Segmentos

Caso inductivo:

$$f.(x \triangleright xs)$$

= [Partición con $as = [] \vee as \neq []$ y prop. de listas]

$$\langle \text{Min } bs, cs : x \triangleright xs = bs + cs : sum.bs \rangle$$

min

$$\langle \text{Min } as, bs, cs : xs = as + bs + cs : sum.bs \rangle$$

Hipótesis inductiva

Introducimos:

$$g.xs = \langle \text{Min } bs, cs : xs = bs + cs : sum.bs \rangle$$

Calcula el la suma del subsegmento inicial con suma mínima

Derivando g

$$g.(x \triangleright xs)$$

= [Partiendo rango $as = [] \vee as \neq []$ y prop. listas]

$$0 \min \langle \text{Min } bs, cs : xs = bs + cs : sum.(x \triangleright bs) \rangle$$

= [dist. + y Min]

$$0 \min (x + \langle \text{Min } bs, cs : xs = bs + cs : sum.bs \rangle)$$

= [Inducción]

$$0 \min (x + g.xs)$$

Es decir nos queda:

$$g : [Num] \mapsto Num$$

$$g.[] \doteq 0$$

$$g.(x \triangleright xs) \doteq 0 \min (x + g.xs)$$

y

$$f : [Num] \mapsto Num$$

$$f.[] \doteq 0$$

$$f.(x \triangleright xs) \doteq (x + g.xs) \min f.xs$$

Mejorando la Solución

Podemos usar tupling para mejorar la solución:

$$h.xs = (f.xs, g.xs)$$

Caso base:

$$\begin{aligned} h.[] &= \{ \text{especificación de } h \} \\ &\quad (f.[], g.[]) \\ &= \{ \text{definición de } f \text{ y } g \} \\ &\quad (0,0) \end{aligned}$$

Mejorando la Solución

Veamos el caso inductivo:

$$h.(x \triangleright xs)$$

= { especificación de h }

$$(f.(x \triangleright xs), g.(x \triangleright xs))$$

= { definición de f y g }

$$((x + g.xs) \min f.xs, 0 \min (x + g.xs))$$

= { introducimos a, b }

$$((x + b) \min a, 0 \min (x + b))$$

$$\|[(a, b) = (f.xs, g.xs)]\|$$

= { hipótesis inductiva }

$$((x + b) \min a, 0 \min (x + b))$$

$$\|[(a, b) = h.xs]\|$$

Nos queda:

$$h : [Num] \mapsto (Num, Num)$$

$$h.[] \doteq (0, 0)$$

$$h.(x \triangleright xs) \doteq ((x + b) \min a, 0 \min (x + b)) \\ \|[(a, b) = h.xs]\|$$

Esta solución es lineal