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:

$f.0$

$= $ [definición]

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

$= $ [Rango vacío]

0

Caso Inductivo:

$f.(n+1)$

$= $ [definición]

$\langle \sum i : 0 \leq i < (n+1) : X^i \rangle$

$= $ [Aritmetica]

$\langle \sum i : 0 \leq i < n \vee n \leq i < n+1 : X^i \rangle$

$= $ [Aritmetica]

$\langle \sum i : 0 \leq i < n \vee i = n : X^i \rangle$

$= $ [Partición de rango y rango único]

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

$= $ [Inducción]

$f.n + X^n$

Modularizando

La función anterior queda:

$$f : Num \to Num$$
$$f.0 \doteq 0$$
$$f.(n+1) \doteq f.n + X^n$$

Para calcular X^n podemos agregar una función local:

$$f : Num \to Num$$
$$f.0 = 0$$
$$f.(n+1) = g.n + f.n$$
$$[\![g.i = X^i]\!]$$

Ejercicio: Derivar g

Otra Derivación

Podemos derivar la función de otra forma:

$f.(n+1)$

$=$ [definición]

$\langle \sum i : 0 \leq i < (n+1) : X^i \rangle$

$=$ [Aritmetica]

$\langle \sum i : 0 \leq i < 1 \vee 1 \leq i < n+1 : X^i \rangle$

$=$ [Partición de rango y Rango único]

$X^0 + \langle \sum i : 1 \leq i < (n+1) : X^i \rangle$

$=$ [Reemplazando i por j+1]

$X^0 + \langle \sum j : 1 \leq j+1 < (n+1) : X^{j+1} \rangle$

$=$ [Aritmética]

$X^0 + \langle \sum j : 1 \leq j+1 < (n+1) : X^j * X \rangle$

$=$ [Prop. Cuantificadores]

$X^0 + X * \langle \sum j : 1 \leq j+1 < (n+1) : X^j \rangle$

$=$ [Aritmética]

$X^0 + X * \langle \sum j : 0 \leq j < n : X^j \rangle$

$=$ [Inducción]

$X^0 + X * f.n$

Hace menos multiplicaciones que antes

Tupling

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

$$
\begin{aligned}
&\mathit{fib} : \mathit{Num} \to \mathit{Num}\\
&\mathit{fib}.0 = 0\\
&\mathit{fib}.1 = 1\\
&\mathit{fib}.(n+2) = \mathit{fib}.(n+1) + \mathit{fib}.n
\end{aligned}
$$

Función fibonacci

Esta definición de fibonacci toma tiempo exponencial.

Mejorando la Eficiencia

Podemos calcular la siguiente función:

$$g : Num \to (Num, Num)$$
$$g.n = (fib.n, fib.(n+1))$$

Para el caso base:

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

Usando Tuplas

Veamos el caso inductivo:

Se recalcula fib.(n+1)

$g.(n+1)$

$=$ Def. g

$(fib.(n+1), fib.(n+2))$

$=$ Def. fib

$(fib.(n+1), fib.n + fib.(n+1))$

$=$ [intro. de a y b]

$(fib.(n+1), fib.n + fib.(n+1))$

$[\![a = fib.n$

$b = fib.(n+1)]\!]$

$=$ [igualdad de pares]

$(fib.(n+1), fib.n + fib.(n+1))$

$[\![(a, b) = (fib.n, fib.(n+1))]\!]$

$=$ [reemplazo]

$(b, a+b)$

$[\![(a, b) = (fib.n, fib.(n+1))]\!]$

$=$ [Inducción]

$(b, a+b)$

$[\![(a, b) = g.n]\!]$

Usando Tuplas

La función queda:

$$
\begin{aligned}
&f : Num \to Num\\
&g.0 = (0, 1)\\
&g.(n+1) = (b, a+b)\\
&\qquad\qquad\quad [\![(a, b) = g.n]\!]
\end{aligned}
$$

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:

$[] \uparrow n = []$
$xs \uparrow 0 = []$
$(x \triangleright xs) \uparrow (n+1) = x \triangleright (xs \uparrow n)$

$$P : [Num] \to Num$$

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

Dice si la suma de todos los subsegmentos iniciales es mayor o igual que cero

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

$$P(x \triangleright xs)$$
$$= [Def.]$$
$$\langle \forall i : 0 \leq i \leq \#(x \triangleright xs) : sum((x \triangleright xs) \uparrow i) \geq 0 \rangle$$
$$= [Part\,.\,Rango]$$
$$0 \geq 0 \wedge \langle \forall i : 0 \leq i \leq \#xs : x + sum(xs \uparrow i) \geq 0 \rangle$$

En este punto no se puede aplicar la hipótesis inductiva

Generalización

Podemos generalizar la **especificación**:

$$Q : Num \to [Num] \to Bool$$
$$Q\,.\,n\,.\,xs = \langle \forall i : 0 \leq i \leq \#xs : n + sum(xs \uparrow i) \geq 0 \rangle$$

Esta función es más general que la original:

$$P\,.\,xs = Q.0.xs$$

Trataremos de derivar esta función. El caso base, por rango vacío obtenemos:

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

Ejercicio!

Generalización

Derivemos Q:

$$
\begin{aligned}
& Q\,.\,n\,.\,(x \triangleright xs) \\
& = [Def] \\
& \langle \forall i : 0 \le i \le 1 + \#xs : n + sum(x \triangleright xs \uparrow i) \ge 0 \rangle \\
& = [Part\,.\,Rango, Def\,.\uparrow, Def\,.\,sum] \\
& n + x \ge 0 \wedge \langle \forall i : 0 \le i \le \#xs : n + sum(xs \uparrow i) \ge 0 \rangle \\
& = [H\,.\,I.] \\
& n + x \ge 0 \wedge Q\,.\,n + x\,.\,xs
\end{aligned}
$$

Es decir, nos queda:

$$Q\,.\,n\,.\,[\,] = n \ge 0$$

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

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

Ejemplos con Subsegmentos

Veamos un ejemplo con subsegmentos.

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

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

Problema del subsegmento de suma minima

El caso base queda como ejercicio, y obtenemos:

$$f.[] = 0$$

Problemas con Segmentos

Caso inductivo:

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

$$= [\text{Partición con } as = [\,] \vee as \neq [\,] \text{ y prop. de listas}]$$

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

$$\text{min}$$

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

Podemos introducir una función nueva para calcular esto

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

$g.(x \triangleright xs)$

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

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

$= [\text{dist. } + \text{ y Min}]$

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

$= [\text{Inducción}]$

$0 \text{ min } (x + g.xs)$

Es decir nos queda:

$$\begin{array}{rcl} g : [Num] \mapsto Num & & \\ g.[\,] & \doteq & 0 \\ g.(x \triangleright xs) & \doteq & 0 \text{ min } (x + g.xs) \end{array}$$

y

$$\begin{array}{rcl} f : [Num] \mapsto Num & & \\ f.[\,] & \doteq & 0 \\ f.(x \triangleright xs) & \doteq & (x + g.xs) \text{ min } f.xs \end{array}$$

Mejorando la Solución

Podemos usar tupling para mejorar la solución:

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

Caso base:

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

Mejorando la Solución

Veamos el caso inductivo:

$$
\begin{aligned}
& h.(x \triangleright xs) \\
= & \ \{ \text{ especificación de } h \ \} \\
& (f.(x \triangleright xs),\ g.(x \triangleright xs)) \\
= & \ \{ \text{ definición de } f \text{ y } g \ \} \\
& ((x + g.xs) \text{ min } f.xs,\ 0 \text{ min } (x + g.xs)) \\
= & \ \{ \text{ introducimos } a,\ b \ \} \\
& ((x + b) \text{ min } a,\ 0 \text{ min } (x + b)) \\
& \qquad \|[(a, b) = (f.xs, g.xs)]\| \\
= & \ \{ \text{ hipótesis inductiva } \} \\
& ((x + b) \text{ min } a,\ 0 \text{ min } (x + b)) \\
& \qquad \|[(a, b) = h.xs]\|
\end{aligned}
$$

Nos queda:

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

$$
\begin{aligned}
h.[\,] &\doteq (0, 0) \\
h.(x \triangleright xs) &\doteq ((x + b) \text{ min } a,\ 0 \text{ min } (x + b)) \\
& \qquad \|[(a, b) = h.xs]\|
\end{aligned}
$$

Esta solución es lineal