Tema 10. Razonamiento ecuacional

10.1 Pruebas directas

- ✓ El razonamiento formal con programas funcionales es simple
- ✓ Gracias a la *transparencia referencial* podemos sustituir términos *equivalentes*
- ✓ Consideraremos las ecuaciones en una definición de función como equivalencias (*Axiomas*)

EJEMPLO

$$\begin{array}{lcl} cambio & :: & (a,b) \to (b,a) \\ cambio\ (x,y) & = & (y,x) \end{array}$$

Demostrar que

$$\forall\ m{::}a,\ n{::}b\ .\ cambio\ (cambio\ (m,n)) \equiv (m,n)$$

Axiomas

$$\forall\ x{::}a,\ \forall\ y{::}b\ .\ cambio\ (x,y) \equiv (y,x) \quad -- \ Ax_1$$

Demostración

$$\begin{array}{ll} & cambio\ (\underline{cambio\ (m,n)}) \\ \equiv & \quad \{\text{por } Ax_1\} \\ & \underline{cambio\ (n,m)} \\ \equiv & \quad \{\text{por } Ax_1\} \\ & (m,n) \end{array}$$

10.2 Pruebas por casos

✓ Para tipos no recursivos, basta con probar la propiedad para cada constructor

EJEMPLO

$$\textbf{data}\ Bool = False \mid True$$

$$\begin{array}{lcl} not & :: & Bool \to Bool \\ not\ True & = & False \\ not\ False & = & True \end{array}$$

Demostrar que

$$\forall\ x{::}Bool\ .\ not\ (not\ x) \equiv x$$

Axiomas

$$\begin{array}{lcll} not\ True & \equiv & False & -- \ Ax_1 \\ not\ False & \equiv & True & -- \ Ax_2 \end{array}$$

Demostración

✓ Si x es $False$, hay que demostrar

$$not\ (not\ False) \equiv False$$

$$\begin{array}{ll} & not\ (\underline{not\ False}) \\ \equiv & \quad \{\text{por } Ax_2\} \\ & \underline{not\ True} \\ \equiv & \quad \{\text{por } Ax_1\} \\ & False \end{array}$$

✓ Si x es $True$, hay que demostrar

$$not\ (not\ True) \equiv True$$

$$\begin{array}{ll} & not\ (\underline{not\ True}) \\ \equiv & \quad \{\text{por } Ax_1\} \\ & \underline{not\ False} \\ \equiv & \quad \{\text{por } Ax_2\} \\ & True \end{array}$$

✓ La propiedad queda demostrada para cualquier $x{::}Bool$

10.3 Pruebas por inducción

✓ Para tipos recursivos, el número de casos a considerar es infinito

✓ No podemos demostrar todos los casos

✓ Se usa la *inducción*

EJEMPLO

$$\textbf{data}\ Nat\ =\ Cero \mid Suc\ Nat\ \ \textbf{deriving}\ Show$$

$$\begin{array}{lll} (<+>) & :: & Nat \to Nat \to Nat \\ m\ <+>\ Cero & = & m \\ m\ <+>\ Suc\ n & = & Suc\ (m\ <+>\ n) \end{array}$$

Demostrar que

$$\forall\ x{::}Nat\ .\ Cero\ <+>\ x\ \equiv\ x$$

Principio de inducción para el tipo Nat

$$\forall x :: Nat\ .\ P(x) \Leftrightarrow \left\{ \begin{array}{l} P(Cero) \\ \wedge \\ \forall x :: Nat\ .\ P(x) \Rightarrow P(Suc\ x) \end{array} \right.$$

✓ **Caso base:** Hay que demostrar $P(Cero)$

✓ **Paso inductivo:** Hay que demostrar $P(Suc\ x)$ supuesto $P(x)$

Pruebas por inducción (2)

Propiedad

$$\forall\, x{::}Nat \;.\; Cero \;<+>\; x \;\equiv\; x$$

Axiomas

$$\begin{array}{lll} \forall\, m{::}Nat & . \; m \;<+>\; Cero \;\equiv\; m & -- \; Ax_1 \\ \forall\, m,\, n{::}Nat & . \; m \;<+>\; Suc\; n \;\equiv\; Suc\; (m \;<+>\; n) & -- \; Ax_2 \end{array}$$

✓ **Caso base:** Hay que demostrar

$$Cero \;<+>\; Cero \;\equiv\; Cero$$

Demostración

$$\begin{array}{ll} & \underline{Cero \;<+>\; Cero} \\ \equiv & \quad \{\text{por } Ax_1\} \\ & Cero \end{array}$$

✓ **Paso inductivo:** Hay que demostrar

$$\forall\, x{::}Nat \;.\; \underbrace{(Cero \;<+>\; x \;\equiv\; x)}_{Hipótesis\ de\ Inducción} \Rightarrow (Cero \;<+>\; Suc\; x \;\equiv\; Suc\; x)$$

Demostración

$$\begin{array}{ll} & \underline{Cero \;<+>\; Suc\; x} \\ \equiv & \quad \{\text{por } Ax_2\} \\ & Suc\; (\underline{Cero \;<+>\; x}) \\ \equiv & \quad \{\text{por hipótesis de inducción}\} \\ & Suc\; x \end{array}$$

✓ La propiedad queda demostrada para cualquier $x{::}Nat$

Pruebas por inducción (3)

✓ Las listas también son un tipo recursivo

$$\textbf{data}\ [a] \ = \ [\,] \ \mid \ a \ : \ [a]$$

Principio de inducción para listas

$$\forall ls :: [a] \ \textbf{.}\ P(ls) \Leftrightarrow \left\{ \begin{array}{l} P([\,]) \\ \wedge \\ \forall xs :: [a], \forall x :: a \ \textbf{.} \ \ P(xs) \Rightarrow P(x : xs) \end{array} \right.$$

✓ **Caso base:** Hay que demostrar $P([\,])$

✓ **Paso inductivo:** Hay que demostrar $P(x : xs)$ supuesto $P(xs)$

EJEMPLO

$$\begin{array}{lcl} suma & :: & [Int] \rightarrow Int \\ suma\ [\,] & = & 0 \\ suma\ (x : xs) & = & x \ + \ suma\ xs \end{array}$$

$$\begin{array}{lcl} doble & :: & [Int] \rightarrow [Int] \\ doble\ [\,] & = & [\,] \\ doble\ (x : xs) & = & 2 * x \ : \ doble\ xs \end{array}$$

Demostrar

$$\forall\ ls{::}[Int]\ \textbf{.}\ suma\ (doble\ ls) \ \equiv \ 2 \ * \ suma\ ls$$

✓ **Caso base:** Hay que demostrar $suma\ (doble\ [\,]) \ \equiv \ 2 \ * \ suma\ [\,]$

✓ **Paso inductivo:** Hay que demostrar

$$\begin{array}{l} \forall\ xs{::}[Int],\ \forall\ x{::}Int\ \textbf{.} \\ \quad suma\ (doble\ xs) \ \equiv \ 2 \ * \ suma\ xs \qquad \text{-- Hipótesis de inducción} \\ \Rightarrow \\ \quad suma\ (doble\ (x : xs)) \ \equiv \ 2 \ * \ suma\ (x : xs) \end{array}$$

Pruebas por inducción (4)

Axiomas

$$\begin{array}{lll} & suma\ [\,] & \equiv\ 0 \qquad\qquad -- \ AxSuma_1 \\ \forall\ x{::}Int,\ \forall\ xs{::}[Int]\ .\ & suma\ (x : xs) & \equiv\ x\ +\ suma\ xs \qquad -- \ AxSuma_2 \end{array}$$

$$\begin{array}{lll} & doble\ [\,] & \equiv\ [\,] \qquad\qquad\ -- \ AxDoble_1 \\ \forall\ x{::}Int,\ \forall\ xs{::}[Int]\ .\ & doble\ (x : xs) & \equiv\ 2 * x\ :\ doble\ xs \qquad -- \ AxDoble_2 \end{array}$$

✓ **Caso base:** Hay que demostrar

$$suma\ (doble\ [\,]) \equiv 2\ *\ suma\ [\,]$$

Demostración

$$\begin{array}{ll} & \underline{suma\ (doble\ [\,])} \\ \equiv & \quad \{\text{por } AxDoble_1\} \\ & \underline{suma\ [\,]} \\ \equiv & \quad \{\text{por } AxSuma_1\} \\ & 0 \end{array}$$

$$\begin{array}{ll} & 2\ *\ \underline{suma\ [\,]} \\ \equiv & \quad \{\text{por } AxSuma_1\} \\ & 2\ *\ 0 \\ \equiv & \quad \{\text{aritmética}\} \\ & 0 \end{array}$$

✓ **Paso inductivo:** Hay que demostrar

$$\begin{array}{l} \forall\ xs{::}[Int],\ \forall\ x{::}Int\ . \\ \quad suma\ (doble\ xs) \equiv 2\ *\ suma\ xs \qquad -- \text{ Hipótesis de inducción} \\ \Rightarrow \\ \quad suma\ (doble\ (x : xs)) \equiv 2\ *\ suma\ (x : xs) \end{array}$$

Demostración

$$\begin{array}{ll} & \underline{suma\ (doble\ (x : xs))} \\ \equiv & \quad \{\text{por } AxDoble_2\} \\ & \underline{suma\ (2 * x\ :\ doble\ xs)} \\ \equiv & \quad \{\text{por } AxSuma_2\} \\ & 2 * x\ +\ \underline{suma\ (doble\ xs)} \\ \equiv & \quad \{\text{por hipótesis de inducción}\} \\ & 2 * x\ +\ 2 * suma\ xs \end{array}$$

$$\begin{array}{ll} & 2\ *\ \underline{suma\ (x : xs)} \\ \equiv & \quad \{\text{por } AxSuma_2\} \\ & \underline{2\ *\ (x\ +\ suma\ xs)} \\ \equiv & \quad \{\text{distributiva de } (*) \text{ y } (+)\} \\ & 2 * x\ +\ 2 * suma\ xs \end{array}$$

Pruebas por inducción (5)

✓ Los árboles binarios son un tipo recursivo

$$\textbf{data}\ \ \acute{A}rbolB\ a\ =\ \ VacíoB \mid NodoB\ (\acute{A}rbolB\ a)\ a\ (\acute{A}rbolB\ a)$$

Principio de inducción para el tipo $\acute{A}rbolB\ a$

$$\forall t :: \acute{A}rbolB\ a\ .\ P(t) \Leftrightarrow \left\{ \begin{array}{l} P(VacíoB) \\ \wedge \\ \forall i, d :: \acute{A}rbolB\ a,\ \forall r :: a\ . \\ \quad P(i) \wedge P(d) \Rightarrow P(NodoB\ i\ r\ d) \end{array} \right.$$

✓ **Caso base:** Hay que demostrar $P(VacíoB)$

✓ **Paso inductivo:** Hay que demostrar $P(NodoB\ i\ r\ d)$ supuestos $P(i)$ y $P(d)$

✓ Los árboles generales son un tipo recursivo

$$\textbf{data}\ \acute{A}rbol\ a\ =\ \ Vacío \mid Nodo\ a\ [\acute{A}rbol\ a]$$

Principio de inducción para el tipo $\acute{A}rbol\ a$

$$\forall t :: \acute{A}rbol\ a\ .\ P(t) \Leftrightarrow \left\{ \begin{array}{l} P(Vacío) \\ \wedge \\ \forall xs :: [\acute{A}rbol\ a],\ \forall r :: a\ . \\ \quad \forall x \in xs\ \ .\ \ P(x) \Rightarrow P(Nodo\ r\ xs) \end{array} \right.$$

✓ **Caso base:** Hay que demostrar $P(Vacío)$

✓ **Paso inductivo:** Hay que demostrar $P(Nodo\ r\ xs)$ supuesto $P(x)$ para todo x perteneciente a xs

Propiedades con varias variables

✓ Si en la propiedad aparecen varias variables, se puede hacer la inducción sobre cualquiera de ellas

✓ Si al intentarlo sobre una concreta la demostración se complica, lo intentamos sobre otra

EJEMPLO

$$
\begin{array}{lcl}
length & :: & [a] \to Int \\
length\ [\,] & = & 0 \\
length\ (x : xs) & = & 1 + length\ xs
\end{array}
$$

$$
\begin{array}{lcl}
(+\!\!+) & :: & [a] \to [a] \to [a] \\
[\,] +\!\!+ ys & = & ys \\
(x : xs) +\!\!+ ys & = & x : (xs +\!\!+ ys)
\end{array}
$$

Demostrar:

$$\forall\ xs,\ ys{::}[a]\ .\ length\ (xs +\!\!+ ys) \equiv length\ xs + length\ ys$$

Axiomas:

$$
\begin{array}{lcll}
 & length\ [\,] \equiv 0 & -- & Ax Length_1 \\
\forall\ x{::}a,\ \forall\ xs{::}[a]\ . & length\ (x : xs) \equiv 1 + length\ xs & -- & Ax Length_2
\end{array}
$$

$$
\begin{array}{lcll}
\forall\ ys{::}[a]\ . & [\,] +\!\!+ ys \equiv ys & -- & Ax Concat_1 \\
\forall\ x{::}a,\ \forall\ xs{::}[a],\ \forall\ ys{::}[a]\ . & (x : xs) +\!\!+ ys \equiv x : (xs +\!\!+ ys) & -- & Ax Concat_2
\end{array}
$$

Propiedades con varias variables (2)

Propiedad

$$\forall\ xs,\ ys{::}[a]\ .\ length\ (xs +\!\!+ ys) == length\ xs + length\ ys$$

✓ Por inducción sobre xs

✓ **Caso base:** Hay que demostrar

$$\forall\ ys{::}[a]\ .\ length\ ([\,] +\!\!+ ys) == length\ [\,] + length\ ys$$

✓ **Paso inductivo:** Hay que demostrar

$$\begin{array}{l}\forall\ xs{::}[a],\ \forall\ x{::}a\ .\\ \quad \forall\ ys{::}[a]\ .\ length\ (xs +\!\!+ ys) == length\ xs + length\ ys\\ \Rightarrow\\ \quad \forall\ ys{::}[a]\ .\ length\ ((x : xs) +\!\!+ ys) == length\ (x : xs) + length\ ys\end{array}$$

✓ Por inducción sobre ys

✓ **Caso base:** Hay que demostrar

$$\forall\ xs{::}[a]\ .\ length\ (xs +\!\!+ [\,]) == length\ xs + length\ [\,]$$

✓ **Paso inductivo:** Hay que demostrar

$$\begin{array}{l}\forall\ ys{::}[a],\ \forall\ y{::}a\ .\\ \quad \forall\ xs{::}[a]\ .\ length\ (xs +\!\!+ ys) == length\ xs + length\ ys\\ \Rightarrow\\ \quad \forall\ xs{::}[a]\ .\ length\ (xs +\!\!+ (y : ys)) == length\ xs + length\ (y : ys)\end{array}$$

Objetivos del tema

El alumno debe:

- ✓ Conocer cómo razonar formalmente con programas funcionales
- ✓ Conocer cómo razonar con tipos no recursivos
- ✓ Conocer cómo razonar con tipos recursivos (principios de inducción)
- ✓ Saber demostrar propiedades usando los métodos anteriores