Inducción

Programación Avanzada
UNRC
Pablo Castro

Inducción Matemática

La inducción matemática nos permite probar propiedades sobre **tipos de datos inductivos**:

Naturales: `data Nat = Zero|Succ Nat`

Listas: `data List a = Nil|Cons a List a`

Árboles: `data Tree a = Nil|Node(Tree a)a(Tree a)`

Todos los tipos que pueden definirse de esta forma se llaman inductivos

Inducción sobre Naturales

La inducción sobre naturales puede escribirse de la siguiente forma:

Si $p : Nat \to Bool$ es un predicado sobre naturales entonces

$$p.0 \wedge \langle \forall n :: p.n \Rightarrow p.(n+1) \rangle \Rightarrow \langle \forall n :: p.n \rangle$$

se cumple para 0

Si se cumple para n entonces se cumple para n+1

Entonces se cumple para todos los naturales

La inducción nos permite probar propiedades sobre los naturales.

Inducción sobre Naturales

En la práctica cuando queremos probar una propiedad por inducción podemos hacer los siguiente:

- Probar el caso base: $p.0$
- Asumir la hipótesis inductiva: $p.n$
- Probar el caso inductivo: $p.(n+1)$

A veces puede haber más de un caso base, estos son los primero valores en los cuales la propiedad vale

Ejemplo

Probemos

$$\langle \forall n :: \langle \sum i : 0 \leq i \leq n : i \rangle = \frac{n * (n+1)}{2} \rangle$$

Caso base:

$P.0$

$= [\text{Def. de } P]$

$\langle \sum i : 0 \leq i \leq 0 : i \rangle = \frac{0 * (0+1)}{2}$

$= [\text{Aritmética}]$

$\langle \sum i : i = 0 : i \rangle = 0$

$= [\text{Rango único}]$

$0 = 0$

$= [\text{Lógica}]$

True

Caso inductivo:

$P.(n+1)$

$= [\text{Def. de } P]$

$\langle \sum i : 0 \leq i \leq n+1 : i \rangle = \frac{n * (n+1)}{2}$

$= [\text{Aritmética}]$

$\langle \sum i : 0 \leq i \leq n \vee i = n+1 : i \rangle = \frac{n * (n+1)}{2}$

$= [\text{Partición de Rango}]$

$\langle \sum i : 0 \leq i \leq n : i \rangle + \langle \sum i : i = n+1 : i \rangle = \frac{n * (n+1)}{2}$

$= [\text{Rango único e Hipótesis Ind.}]$

$\frac{n(n+1)}{2} + n + 1 == \frac{(n+1) * (n+2)}{2}$

$= [\text{Aritmética}]$

True

Inducción Fuerte

Una variante al principio de inducción es el de inducción fuerte:

$$p.0 \wedge \langle\langle \forall i : i < n : p.i \rangle \Rightarrow p.n \rangle \Rightarrow \langle \forall n :: p.n \rangle$$

Vale para todos los anteriores

Probemos:

$$\langle \forall n : 2 \leq n : \langle \exists p : prime.p \wedge p \leq n : p \mid n \rangle\rangle$$

Para todo número mayor que dos existe un primo que lo divide

Ejemplo

Caso base:

$P.(2)$

$= [\text{Def. de } P]$

$\langle \exists p : prime.p \wedge 2 \leq p \leq 2 : p \mid 2 \rangle$

$= [\text{Aritmética}]$

$\langle \exists p : prime.p \wedge p = 2 : p \mid 2 \rangle$

$= [\text{Rango único}]$

$2 \mid 2$

$= [\text{Reflex.}|]$

$true$

Caso Inductivo:

$P.(n)$

$= [\text{Def. de } P]$

$\langle \exists p : prime.p \wedge 2 \leq p \leq n : p \mid n \rangle$

$= [\neg primo.n \text{ y Leibniz}]$

$\langle \exists p : prime.p \wedge 2 \leq p \leq n : p \mid (n_1 * n_2) \rangle$

$= [\text{Hipótesis Ind. y } p \mid n_1 \Rightarrow p \mid n_1 * n_2]$

$true$

Caso que n no sea primo, si n es primo es trivial

Ejercicio: Definir prime en Haskell.

Inducción sobre Listas

Para listas el principio de inducción es el siguiente:

$$p.[] \wedge \langle \forall x, xs : x \in A \wedge xs \in [A] : p.xs \Rightarrow p.(x \triangleright xs) \rangle \Rightarrow (\forall xs : xs \in [A] : p.xs)$$

Caso base

Si vale para una lista vale para la lista cuando le agregamos un elemento

Entonces vale para todas las listas

Ejercicio, demostrar:

$$\langle \forall xs, ys, zs :: xs ++ (ys ++ zs) = (xs ++ ys) ++ zs \rangle$$

Inducción y Recursión

Los conceptos de recursión e inducción están fuertemente relacionados:

$$f.0 \doteq 1$$

$$f.1 \doteq 1$$

$$f.(n+2) \doteq f.(n+1) + f.n$$

Implementación recursiva de fibonacci

Supongamos que queremos demostrar la siguiente propiedad del programa:

$$\langle \forall n : 2 \leq n : f.n < 2^n \rangle$$

Inducción y Recursión

El caso base es dejado como ejercicio, veamos el caso inductivo:

$$f.(m+2)$$
$$= [\text{Def. de } f]$$
$$f.(m+1) + f.m$$
$$< [\text{Hip.Inductiva}]$$
$$2^{m+1} + 2^{m}$$
$$< [\text{Aritmética}]$$
$$2^{m+1} + 2^{m+1}$$
$$= [\text{Aritmética}]$$
$$2^{m+2}$$

Otro Ejemplo

Supongamos la función:

$$f : Num \to Num$$
$$f.0 = 0$$
$$f.(n+1) = f.n + 2 * n + 1$$

Podemos demostrar la siguiente propiedad:

$$\langle \forall n :: f.n = n^2 \rangle$$

Demostrar por inducción

Derivando Código

Podríamos haber utilizado la especificación para **derivar** la implementación de la función:

Especificación

$$\langle \forall n :: f.n = n^2 \rangle$$

Caso base:

$$f.0$$
$$= [\text{Especificación}]$$
$$0^2$$
$$=$$
$$0$$

Caso inductivo:

$$f.(n+1)$$
$$= [\text{Especificación}]$$
$$(n+1)^2$$
$$= [\text{Aritmética}]$$
$$(n+1) * (n+1)$$
$$= [\text{Binomios}]$$
$$n^2 + 2 * n + 1$$
$$= [\text{Inducción}]$$
$$f.n + 2 * n + 1$$

Suponemos que la función cumple la especificación para n