### 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

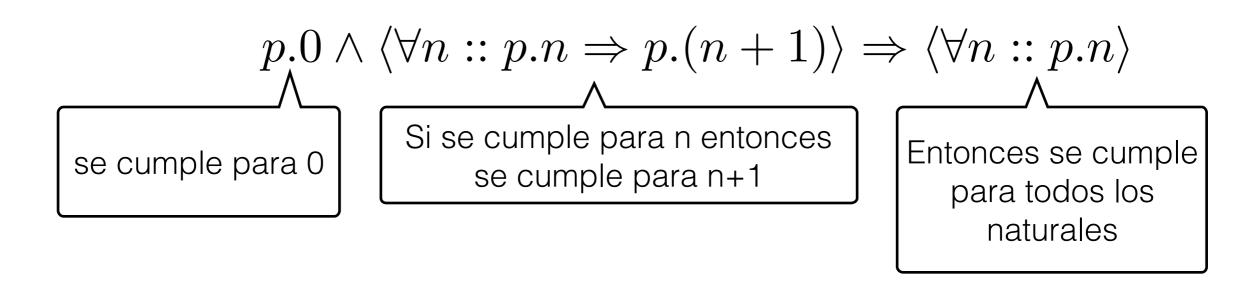
**Árboles:** data Tree a = Nil|Node(Tree a)a(Tree a)

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



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

#### Caso base:

#### P.0

= [Def. de *P*]

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

= [Aritmética]

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

= [Rango único]

$$0 = 0$$

= [Lógica]

True

#### Caso inductivo:

True

$$P.(n+1)$$
= [Def. de P]
$$\langle \sum i : 0 \le i \le n+1 : i \rangle = \frac{n*(n+1)}{2}$$
= [Aritmética]
$$\langle \sum i : 0 \le i \le n \lor i = n+1 : i \rangle = \frac{n*(n+1)}{2}$$
= [Partición de Rango]
$$\langle \sum i : 0 \le i \le n : i \rangle + \langle \sum i : i = n+1 : i \rangle = \frac{n*(n+1)}{2}$$
= [Rango único e Hipótesis Ind.]
$$\frac{n(n+1)}{2} + n + 1 = \frac{(n+1)*(n+2)}{2}$$
= [Aritmética]

### Inducción Fuerte

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

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

Vale para todos los anteriores

Probemos:

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

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

# Ejemplo

#### Caso base:

```
P.(2)
= [Def. de P]
\langle \exists p : prime.p \land 2 \le p \le 2 : p \mid 2 \rangle
= [Aritmética]
\langle \exists p : prime.p \land p = 2 : p \mid 2 \rangle
= [Rango único]
2 \mid 2
= [Reflex.|]
true
```

#### Caso Inductivo:

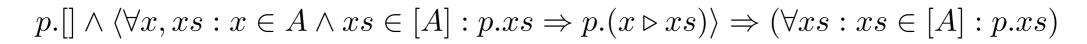
```
P.(n)
= [Def. de P]
\langle \exists p : prime.p \land 2 \le p \le n : p \mid n \rangle
= [\neg primo.n \text{ y Leibniz}]
\langle \exists p : prime.p \land 2 \le p \le n : p \mid (n_1 * n_2) \rangle
= [Hipótesis Ind. y p \mid n_1 \Rightarrow p \mid n_1 * n_2]
true
```

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:



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:

$$f.(m + 2)$$

$$= [Def. de f]$$
 $f.(m + 1) + f.m$ 

$$< [Hip.Inductiva]$$
 $2^{m+1} + 2^m$ 

$$< [Aritmética]$$
 $2^{m+1} + 2^{m+1}$ 

$$= [Aritmética]$$
 $2^{m+2}$ 

## 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:

$$f.0$$
= [Especificación]
 $0^2$ 
=  $0$ 

Caso inductivo:  $f \cdot (n+1)$ 

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