

# Fundamentos de Lenguajes de Programación

José Raymundo Marcial Romero<sup>1</sup>  
José de Jesús Lavalle Martínez<sup>1</sup>

<sup>1</sup>Facultad de Ciencias de la Computación  
BUAP



## ¿Para que sirve la inducción?

La inducción es una técnica que sirve para hacer razonar y trabajar con colecciones de objetos los cuales son:

- *estructurados* de alguna forma bien definida,
- finitos pero de cualquier tamaño y complejos.

La inducción utiliza la naturaleza finita y estructurada de los objetos para tratar la parte compleja.

# Se puede utilizar inducción para ...

... razonar sobre cosas como:

- *números naturales*: cada uno es finito, pero los números naturales pueden ser de cualquier tamaño.
- *Estructuras de datos* como listas, árboles entre otras.
- *programas* en un lenguaje de programación: se pueden escribir programas de cualquier tamaño (pero finitos).
- *derivaciones* de afirmaciones semánticas como  $E \Downarrow 4$ : estas derivaciones son árboles finitos de axiomas y reglas.

# Demostraciones con Inducción Matemática

Sea  $P(-)$  una propiedad de los números naturales. El principio de inducción matemática establece que si

$$P(0) \wedge [\forall k.P(k) \Rightarrow P(k + 1)]$$

se cumple, entonces

$$\forall n.P(n)$$

se cumple.

$k$  es el **parámetro de inducción**.

# Escribir una prueba por inducción

Para demostrar que  $P(n)$  se cumple para todo natural  $n$ , se deben hacer dos cosas:

**Caso base:** demostrar que  $P(0)$  se cumple,

**Paso inductivo:** sea  $k$  un número arbitrario, y se asume que  $P(k)$  se cumple. Esta aseveración se conoce como la hipótesis inductiva o IH, con parámetro  $k$ . Empleando esta aseveración, se demuestra que  $P(k + 1)$  se cumple.

## Varios pasos de evaluación

Demostrar que

$$\sum_{i=0}^n i = \frac{n^2 + n}{2}$$

# Definición por inducción

Se puede definir una función  $f$  sobre los números naturales de la siguiente forma:

**Caso base:** dar un valor para  $f(0)$  directamente.

**Paso inductivo:** dar un valor para  $f(k + 1)$  en términos de  $f(k)$ .

# Definición inductiva de la función factorial

- $fact(0) = 1$ .
- $fac(k + 1) = (k + 1) \times fact(k)$ .

## Reducciones Multi-pasos en $Exp$

- $E \rightarrow^0 E$  para cada expresión  $E$  en  $Exp$
- $E \rightarrow^{k+1} E'$  si existe algún  $E''$  tal que
  - $E \rightarrow^k E''$
  - y  $E'' \rightarrow E'$

Una gramática para representar números naturales

$$N ::= \mathbf{zero} \mid \mathbf{succ}(N).$$

Si se quiere demostrar que una propiedad  $P$  se cumple para cualquier cadena generada de la gramática se debe:

**Caso base:** demostrar que  $P(\mathbf{zero})$  se cumple,

**Paso inductivo:** La HI es que  $P(K)$  se cumple para algún  $K$ .  
Asumiendo la HI, demostrar que  $P(\mathbf{succ}(K))$  se cumple.

**Conclusión**  $P(N)$  es verdadero para cada número  $N$ .

# Definición de funciones en la gramática anterior

Para definir una función  $f$  en la gramática anterior se debe:

**Caso base:** dar el valor para  $f(\mathbf{zero})$  directamente,

**Paso inductivo:** definir  $f(\mathbf{succ}(N))$ , usando (si es necesario)  
 $f(N)$

Por lo tanto,  $f(T)$  queda definida para toda expresión  $N$  que se genere de la gramática.

- Definir una función que convierta un valor de tipo  $N$  a un valor entero es decir del tipo

$$nat2int :: N \rightarrow \mathbb{Z}$$

- Definir una función que convierta un valor de tipo  $\mathbb{Z}$  a un valor de tipo  $N$  es decir

$$int2nat :: \mathbb{Z} \rightarrow N$$

- Definir la función que sume dos expresiones de tipo  $N$  y devuelva el valor en tipo  $N$  es decir

$$add :: (N, N) \rightarrow N$$

- Defina funciones para la resta y multiplicación de expresiones de tipo  $N$  es decir

$$\text{resta} :: (N, N) \rightarrow N$$

$$\text{multiplica} :: (N, N) \rightarrow N$$

Recuerde que en una función como tipo  $(N N) \rightarrow N$  los dos primeros valores de  $N$  se toman como argumentos (es decir el dominio de la función) y el tercer  $N$  como el rango. Por ejemplo de las definiciones se podría obtener que:

$$\text{resta}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{zero}))))), \text{succ}(\text{succ}(\text{zero}))) = \\ \text{succ}(\text{succ}(\text{zero}))$$

Lo anterior demostraría que cada forma de construir un número preserva la propiedad  $P$ , y que si  $P$  es verdadero para el constructor básico **zero**, también lo es para cada número.

**bTree ::= Node | Branch(bTree, bTree)**

Note la similaridad con las expresiones aritméticas.

# Inducción Estructural en árboles binarios

Para demostrar una propiedad  $P(-)$  para todo árbol binario, se deben hacer dos cosas:

**Caso base:** demostrar que  $P(\mathbf{Node})$  se cumple,

**Paso inductivo:** La HI es que  $P(T_1)$  y  $P(T_2)$  se cumplen para árboles arbitrarios  $T_1$  y  $T_2$ . Asumiendo la HI, demostrar que  $P(\mathbf{Branch}(T_1, T_2))$  se cumple.

La conclusión es que  $P(T)$  es verdadera para todo árbol  $T$ .

# Definición de funciones en árboles binarios

Para definir una función  $f$  en árboles binarios se debe:

**Caso base:** dar el valor para  $f(\mathbf{Node})$  directamente,

**Paso inductivo:** definir  $f(\mathbf{Branch}(\mathbf{T}_1, \mathbf{T}_2))$ , usando (si es necesario)  $f(\mathbf{T}_1)$  y  $f(\mathbf{T}_2)$ .

Por lo tanto,  $f(T)$  se define para todo árbol  $T$ .

- Definir la función *hojas* que regresa el número de Nodos que son hojas en un árbol.
- Definir una función llamada *Ramifica* que cuente el número de nodos de la forma **Branch**( $-$ ,  $-$ ) en un árbol.
- Demostrar por inducción en la estructura de un árbol que:

$$\text{hojas}(T) = \text{Ramifica}(T + 1) \text{ para cualquier árbol } T$$

- Un camino de un nodo  $n_1$  a un nodo  $n_k$  en un árbol es una secuencia de nodos  $n_2, n_3, \dots, n_k$  de tal forma que  $n_i$  es padre de  $n_{i+1}$  para  $i = 1, 2, \dots, k$ .
- La longitud de un camino es uno menos que el número de nodos del camino.
- La profundidad de un nodo es la longitud del camino único de la raíz al nodo
- La profundidad de un árbol es la profundidad de la hoja más profunda.
- Un árbol binario perfecto es aquel en el que todas sus hojas están a la misma profundidad (distancia desde la raíz)

Para los siguientes ejercicios, puede utilizar la siguiente gramática sobre árboles binarios para que las definiciones anteriores tengan sentido

$$btree1 ::= Hoja \mid Nodo(btree1, btree1)$$

o bien den ustedes su propia definición de árbol.

- Definir una función que regrese la profundidad de un árbol
- Definir una función que devuelva True si el árbol de entrada es perfecto y False en otro caso.

- Dar una gramática para una lista simplemente ligada.
- Presentar una definición inductiva para determinar la longitud de una lista (*longitud*).
- Presentar una definición inductiva para regresar el primer elemento de una lista (*Primer*).
- Definir la función que elimine el primer elemento de una lista (*Eliminar*).
- Demostrar que

$$\text{longitud}(\text{Eliminar}(L)) = \text{longitud}(L) - 1 \text{ para toda lista } L$$