



# Algoritmos y estructuras de datos II

## Inducción estructural

---

Carlos Gustavo Lopez Pombo  
(Charlie)

Departamento de Computación,  
Facultad de ciencias exactas y naturales,  
Universidad de Buenos Aires





# Inducción

## ¿Qué es la inducción?

Desde un punto de vista **histórico** es un método científico que se basa en **derivar reglas generales a partir observaciones particulares**.





# Inducción

## ¿Qué es la inducción?

Desde un punto de vista **matemático** es un “método” de prueba que se basa en **derivar reglas generales** a partir probar que:

*Si una propiedad vale para un elemento, entonces esto implica que vale para todo otro que lo contenga estrictamente. (aprox)*





# Inducción

**¿Qué es la inducción?**

Repasemos inducción sobre los naturales à *la Algebra I*

$$(\forall n : nat) \left( \sum_{i=0}^n i = \frac{n * (n + 1)}{2} \right)$$





# Inducción

**¿Qué es la inducción?**

**¿Cómo sería con nuestras axiomatizaciones?**

$$(\forall n : \text{nat}) (\text{prod}(\text{suc}(\text{suc}(0)), \text{sumatoria}(0, n)) = \text{prod}(n, \text{suc}(n)))$$





# Inducción

## ¿Qué es la inducción?

suma: nat x nat -> nat

(su\_1) suma(n, 0) = n

(su\_2) suma(n, suc(m)) = suc(suma(n, m))

prod: nat x nat -> nat

(prod\_1) prod(n, 0) = 0

(prod\_2) prod(n, suc(m)) = suma(prod(n, m), n)

sumatoria: nat n x nat m -> nat (n <= m)

(suma\_1) sumatoria(n, 0) = 0

(suma\_2) sumatoria(n, suc(m)) = if (n = m)

then n

else suma(sumatoria(n, m), suc(m))

fi





# Inducción

¿Qué es la inducción?

Vamos con una más fácil...

$$(\forall n, m, p : \text{nat}) (\text{suma}(n, \text{suma}(m, p)) = \text{suma}(\text{suma}(n, m), p))$$





# Inducción

**¿Y por qué funciona?**

*Si una propiedad vale para un elemento, entonces esto implica que vale para todo otro que lo contenga estrictamente. (aprox)*

**Para que esto “funcione” debe tratarse de un conjunto bien ordenado**





# Inducción

**¿Y por qué funciona?**

Se dice que un conjunto  $S$  está **bien ordenado** si existe una relación binaria  $R$  sobre  $S$  tal que:

- (a) ordena totalmente a los elementos, y
- (b) es una relación bien fundada





# Inducción

**¿Y por qué funciona?**

Se dice que una relación binaria sobre un conjunto  $S$  es **bien fundada** si para todo subconjunto  $S'$  de  $S$  existe un elemento mínimo  $o$  tal que para todo otro elemento  $t$  de  $S'$ , el par  $\langle t, o \rangle$  no está en la relación.





# Inducción

## ¿Y por qué funciona?

Tomemos entonces cualquier tipo definido usando el lenguaje de TADs con el que trabajamos definido sobre los generadores  $g_1, \dots, g_n$ ,  $>$  un orden arbitrario entre ellos de forma que los generadores no recursivos sean menores que los recursivos, la relación  $>$  entre términos se define como:

$$g_{ik}(\dots(g_{il}(\text{args}_{il}), \dots, \text{args}_{ik})) > g'_{ij}(\dots(g'_{il}(\text{args}_{il}), \dots, \text{args}_{ij}))$$

$$\text{sii} \quad k \geq j \text{ y existe } i' \text{ tal que } i' \leq t \leq i \text{ tal que para todo } i' \leq l < t, g_{il} = g'_{il} \text{ y } g_{i(l+1)} > g'_{i'(l+1)}$$

En pocas palabras, dado un orden entre los generadores, extendemos ese orden a secuencias finitas de generadores, es decir, si pensamos a los generadores como “letras”, construimos el “orden lexicográfico” de términos.





# Inducción

**¿Y por qué funciona?**

**Lema:** para todo par de términos  $t, t'$  distintos, o bien  $t > t'$  o bien  $t' > t$ .





# Inducción

**¿Y por qué funciona?**

**Lema:** existe un elemento mínimo.

**Demo:** trivial pues se toma como dicho elemento al generador no recursivo que sea menor bajo la relación  $>$ .





# Inducción

**¿Y por qué funciona?**

**Teorema:** los conjuntos de términos contruidos con el lenguaje de TADs con el que trabajamos son conjuntos bien ordenados.





# Inducción

**¿Y por qué funciona?**

**Corolario:** los conjuntos bien ordenados no poseen cadenas (respecto de  $>$ ) descendentes infinitas.





# Inducción

## ¿Y por qué funciona?

**Copado:** (a) las recursiones terminan siempre que tengan bien definidos los casos base y que los casos recursivos que toman un término  $t$  se resuelvan en función de un término  $t'$  tal que  $t > t'$ , y (b) tiene sentido pensar que si demuestro que una propiedad vale para los casos base, y que si vale para un término, entonces vale para los términos que lo contienen.





# Inducción

## Esquema de inducción

Sea  $P$  una fórmula con una única variable libre  $x$  de tipo  $T$ , luego, si queremos ver que  $P$  vale para todo elemento de  $T$ , debemos probar:

$$(\forall x : T)P(x)$$





# Inducción

## Esquema de inducción

Sea  $T$  un tipo con generadores:

$c_1: \text{args} \rightarrow T$

$r_1: T \times \text{args} \rightarrow T$

...

$c_n: \text{args} \rightarrow T$

$r_m: T \times \text{args} \rightarrow T$

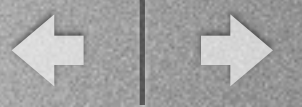
y  $P$  una propiedad con una variable libre  $t$  de tipo  $T$

$$\left( \bigwedge_{i=1}^n (\forall \text{args}) P(c_i(\text{args})) \wedge \bigwedge_{i=1}^m (\forall t : T) (P(t) \Rightarrow (\forall \text{args}) P(r_i(t, \text{args}))) \right)$$

$\Rightarrow$

$$(\forall t : T) P(t)$$





# Inducción

## Esquema de inducción

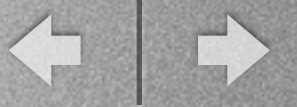
En el caso de los números naturales obtenemos:

$$P(0) \wedge (\forall n : \text{nat})(P(n) \Rightarrow P(\text{suc}(n)))$$

$$\Rightarrow$$

$$(\forall n : \text{nat})P(n)$$





# Inducción

## Esquema de inducción

En el caso de las listas de números naturales obtenemos:

$$P([\ ]) \wedge (\forall l : \text{lista}[\text{nat}]) (P(l) \Rightarrow (\forall n : \text{nat}) P(n \bullet l))$$

$\Rightarrow$

$$(\forall l : \text{lista}[\text{nat}]) P(l)$$





# Repaso

- Presentamos el concepto de inducción generalizando el método conocido desde Álgebra I
- Básicamente sólo eso :-)





¡Es todo por hoy!

