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*

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

Inducción

¿Qué es la inducción?

¿Cómo sería con nuestras axiomatizaciones?

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

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...

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

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 esta **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 <*t*, *o*> 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 g1, ... gn, > 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(...(g_i1 (args_i1), ..., args_ik) > g'_i'j(…(g'_i'1 (args_i'1), ..., args_i'j)

sii k >= j y existe i'1<=t<=i'j tal que para todo i'1<=l<t, g_il = g'_i'l 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 construidos 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:

$$(\forall x : \mathtt{T})\mathrm{P}(x)$$

Inducción

Esquema de inducción

Sea T un tipo con generadores:

c1: args -> T   r1:T t x args -> T

...

cn: args -> T   rm:T t x args -> T

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

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

$$\Rightarrow$$

$$(\forall t : \mathtt{T})\mathtt{P}(t)$$

Inducción

Esquema de inducción

En el caso de los números naturales obtenemos:

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

$$\Rightarrow$$

$$(\forall n : \mathtt{nat})\mathtt{P}(t)$$

Inducción

Esquema de inducción

En el caso de las listas de números naturales obtenemos:

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

$$\Rightarrow$$

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

Repaso

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

¡Es todo por hoy!