Algoritmos y Estructuras de Datos II

Práctica 2 – Demostración de propiedades

Notas preliminares

- En las demostraciones por inducción estructural, justifique **todos** los pasos: por qué axioma, por qué lema, por qué puede aplicarse la hipótesis inductiva, etc. Es importante escribir el **esquema de inducción**, planteando claramente el caso base (o los casos base) y el paso inductivo, e identificando la hipótesis inductiva y la tesis inductiva.
- El alcance de todos los cuantificadores que se utilicen debe estar claramente definido (no omitir paréntesis).
- Demuestre todas las propiedades auxiliares (lemas) que utilice.
- Los ejercicios marcados con el símbolo ★ constituyen un subconjunto mínimo de ejercitación. Sin embargo, aconsejamos fuertemente hacer todos los ejercicios.

Resolver los ejercicios de esta práctica considerando las siguientes definiciones:

Long : secu(α) $\longrightarrow$ nat
- l_1) Long(<>) $\equiv$ 0
- l_2) Long(a $\bullet$ s) $\equiv$ Long(s) + 1

Duplicar : secu(α) $\longrightarrow$ secu(α)
- d_1) Duplicar(<>) $\equiv$ <>
- d_2) Duplicar(a $\bullet$ s) $\equiv$ a $\bullet$ a $\bullet$ Duplicar(s)

$\bullet \circ \bullet$: secu(α) $\times$ α $\longrightarrow$ secu(α)
- $\circ_1$) <> $\circ$ b $\equiv$ b $\bullet$ <>
- $\circ_2$) $(a \bullet s) \circ b$ $\equiv$ a $\bullet$ $(s \circ b)$

$\bullet$ & $\bullet$: secu(α) $\times$ secu(α) $\longrightarrow$ secu(α)
- $\&_1$) <> & t $\equiv$ t
- $\&_2$) $(a \bullet s)$ & t $\equiv$ a $\bullet$ (s & t)

Reverso : secu(α) $\longrightarrow$ secu(α)
- r_1) Reverso(<>) $\equiv$ <>
- r_2) Reverso(a $\bullet$ s) $\equiv$ Reverso(s) $\circ$ a

$\bullet[\bullet]$: secu(α) s $\times$ nat i $\longrightarrow$ α $\{1 \leq i \leq \text{Longitud}(s)\}$
- [ ]) $(a \bullet s)[i]$ $\equiv$ **if** $i = 1$ **then** a **else** $s[i-1]$ **fi**

Tomar : secu(α) $\times$ nat $\longrightarrow$ secu(α)
- t_1) Tomar(<>, n) $\equiv$ <>
- t_2) Tomar($a \bullet s, n$) $\equiv$ **if** $n = 0$ **then** <> **else** a $\bullet$ Tomar($s, n-1$) **fi**

Sacar : secu(α) $\times$ nat $\longrightarrow$ secu(α)
- s_1) Sacar(<>, n) $\equiv$ <>
- s_2) Sacar(a $\bullet$ s, n) $\equiv$ **if** $n = 0$ **then** a $\bullet$ s **else** Sacar($s, n-1$) **fi**

EstáOrdenada? : secu(α) $\longrightarrow$ bool
- E_1) EstáOrdenada?(<>) $\equiv$ true
- E_2) EstáOrdenada?(a $\bullet$ s) $\equiv$ $(a \leq s)$ $\wedge$ EstáOrdenada?(s)

InsertarOrdenado : α $\times$ secu(α) $\longrightarrow$ secu(α) $\{\alpha$ tiene orden total a través de la operación $\leq_\alpha\}$
- O_1) InsertarOrdenado(e, <>) $\equiv$ e $\bullet$ <>

O_2) InsertarOrdenado$(e, a \bullet s)$ $\equiv$ **if** $e \leq_\alpha a$ **then** e • a • s **else** a • InsertarOrdenado(e, s) **fi**

$\bullet \leq \bullet$: $\alpha \times$ secu(α) $\longrightarrow$ bool {α tiene orden total a través de la operación $\leq_\alpha$}
 $\leq_1$) $b \leq <>$ $\equiv$ true
 $\leq_2$) $b \leq a \bullet s$ $\equiv$ $b \leq_\alpha a \wedge b \leq s$

$\bullet < \bullet$: $\alpha \times$ secu(α) $\longrightarrow$ bool {α tiene orden total a través de la operación $<_\alpha$}
 $<_1$) $b < <>$ $\equiv$ true
 $<_2$) $b < a \bullet s$ $\equiv$ $b <_\alpha a \wedge b < s$

$\bullet \leq \bullet$: secu$(\alpha) \times$ secu(α) $\longrightarrow$ bool {α tiene orden total a través de la operación $<_\alpha$ y tiene igualdad a través de la operación $=_\alpha$}
 $\leq s_1$) $<> \leq t$ $\equiv$ true
 $\leq s_2$) $a \bullet s \leq t$ $\equiv$ $\neg$vacia$(t) \wedge (a <_\alpha$ prim$(t) \vee (a =_\alpha$ prim$(t) \wedge s \leq$ fin$(t)))$

h : ab(α) $\longrightarrow$ nat
 h_1) h(nil) $\equiv$ 0
 h_2) h(bin(i, e, d)) $\equiv$ máx(h(i), h(d)) + 1

#Nodos : ab(α) $\longrightarrow$ nat
 $\#n_1$) #Nodos(nil) $\equiv$ 0
 $\#n_2$) #Nodos(bin(i, e, d)) $\equiv$ #Nodos(i) + #Nodos(d) + 1

#Hojas : ab(α) $\longrightarrow$ nat
 $\#h_1$) #Hojas(nil) $\equiv$ 0
 $\#h_2$) #Hojas(bin(i, e, d)) $\equiv$ **if** nil?$(i) \wedge$ nil?(d) **then** 1 **else** #Hojas(i) + #Hojas(d) **fi**

#Internos : ab(α) $\longrightarrow$ nat
 $\#i_1$) #Internos(nil) $\equiv$ 0
 $\#i_2$) #Internos(bin(i, e, d)) $\equiv$ **if** nil?$(i) \wedge$ nil?(d) **then**
  0
 else
  #Internos(i) + #Internos(d) + 1
 fi

def? : clave $\times$ dicc(clave, sign) $\longrightarrow$ bool
 def_1) def?$(c$,vacío) $\equiv$ false
 def_2) def?$(c$,definir(k, s, d)) $\equiv$ $(c =_{\text{clave}} k) \vee$ def?(c, d)

borrar : dicc(clave, sign) d $\times$ clave c $\longrightarrow$ dicc(clave, sign) {def?(d, c)}
 b_1) borrar$(k$, definir(c, s, d)) $\equiv$ **if** $c =_{\text{clave}} k$ **then**
  if def?(k, d) **then** borrar(k, d) **else** d **fi**
 else
  definir$(c, s$, borrar(k, d))
 fi

Ejercicio 1

Demuestre que "$(\forall\ s : \text{secu}(\alpha))$ (Long(Duplicar(s)) $\equiv 2 *$ Long(s))".

Ejercicio 2

Demuestre que "$(\forall\ s : \text{secu}(\alpha))$ (Long(Reverso(s)) $\equiv$ Long(s))".

Ejercicio 3

Demuestre que "$(\forall\ s, t : \text{secu}(\alpha))$ (Long$(s$ & $t)$ $\equiv$ Long(s) + Long(t))".

Ejercicio 4 ★

Demuestre que "($\forall\ s, t$: secu(α)) (Reverso(s & t) $\equiv$ Reverso(t) & Reverso(s))".

Ejercicio 5 ★

Demuestre que "($\forall\ r, s, t$: secu(α)) (($s \leq t \equiv$ true) $\Rightarrow$ (((r & s) $\leq$ (r & t)) $\equiv$ true))".

Puede utilizar, sin necesidad de demostrar, la siguiente propiedad

"($\forall\ s, t$: secu(α)) ($s \leq t \equiv$ true $\Rightarrow$ ($\forall\ a : \alpha$) ($a\ \bullet s \leq a\ \bullet t \equiv$ true))".

Ejercicio 6 ★

Demuestre que "($\forall\ s, t$: secu(α)) ($\forall\ i$: nat)
(($1 \leq i \leq$ Long(s)+Long(t) $\equiv$ true) $\Rightarrow$ ((s&t)[i] $\equiv$ **if** ($1 \leq i \leq$ Long(s)) **then** $s[i]$ **else** $t[i-$Long(s)] **fi**))".

Ejercicio 7 ★

Demuestre que "($\forall\ s$: secu(α)) ($\forall\ i$: nat)
(($1 \leq i \leq$ Long(s) $\equiv$ true) $\Rightarrow$ (Reverso(s)[i] $\equiv s$[Long(s) $- i + 1$]))".

Ejercicio 8 ★

Demuestre que "($\forall\ s$: secu(α)) ($\forall\ n$: nat) (Tomar(s, n) & Sacar(s, n) $\equiv s$)" de dos maneras. La primera por inducción en s y la segunda por inducción en n.

Ejercicio 9 ★

Demuestre que *InsertarOrdenado* es correcto con respecto a *EstáOrdenada?*, es decir, que
"($\forall\ s$: secu(α)) ($\forall\ c : \alpha$)
((EstáOrdenada?(s) $\equiv$ true) $\Rightarrow$ (EstáOrdenada?(InsertarOrdenado(c, s)) $\equiv$ true))".

Ayuda: Demuestre primero las siguientes propiedades:

1. "($\forall\ s$: secu(α)) ($\forall\ a, b : \alpha$) (($a \leq_\alpha b \wedge b \leq s$) $\equiv$ true $\Rightarrow$ ($a \leq s$) $\equiv$ true)".
2. "($\forall\ s$: secu(α)) ($\forall\ a, b : \alpha$)
(EstáOrdenada?(s) $\equiv$ true $\Rightarrow$ ($a \leq$ InsertarOrdenado(b, s) $\equiv$ ($a \leq_\alpha b$) $\wedge$ ($a \leq s$)))".

Ejercicio 10 ★

Demuestre que "($\forall\ a$: ab(nat)) (#Nodos(a) $\leq 2^{\mathrm{h}(a)} - 1 \equiv$ true)".

Ayuda: Puede asumir la propiedad "($\forall\ x, y$: nat) ($2^x + 2^y \leq 2^{\text{máx}(x,y)+1} \equiv$ true)" sin necesidad de probarla.

Ejercicio 11 ★

Demuestre que "($\forall\ a$: ab(α)) (#Hojas(a) $\leq$ #Internos(a) + 1 $\equiv$ true)".

Ejercicio 12 ★

Demuestre que

"($\forall$ a: ab(α)) (Inorder(a) $\equiv$ Reverso(Inorder(Espejo(a))))",

donde

Inorder : ab(α) $\longrightarrow$ secu(α)

i_1)	Inorder(nil)	$\equiv$ <>
i_2)	Inorder(bin(i, e, d))	$\equiv$ Inorder(i) & e $\bullet$ Inorder(d)

Espejo : ab(α) $\longrightarrow$ ab(α)

e_1)	Espejo(nil)	$\equiv$ nil
e_2)	Espejo(bin(i, e, d))	$\equiv$ bin(Espejo(d), e,Espejo(i))

Ejercicio 13

Dada la siguiente especificación de relaciones binarias entre naturales, demuestre que
"($\forall\ R$: relbin) ($\forall\ x, y$: nat) (rel?(x, R, y) $\equiv y \in$ relacionadosXDer(x, R))".

TAD RelBin

exporta relbin, rel?, $\emptyset$, rel, relacionadosXDer

usa Nat, Bool, Conj(nat, $=_{\text{nat}}$)

géneros relbin

igualdad observacional

$(\forall r_1, r_2 : \text{relbin})\ \left(r_1 =_{\text{obs}} r_2 \Longleftrightarrow ((\forall n_1, n_2 \text{: nat})\ (\text{rel?}(n_1, r_1, n_2) =_{\text{obs}} \text{rel?}(n_1, r_2, n_2)))\right)$

observadores básicos

rel? : nat × relbin × nat $\longrightarrow$ bool

generadores

$\emptyset$: $\longrightarrow$ relbin

rel : nat × relbin × nat $\longrightarrow$ relbin

otras operaciones

relacionadosXDer : nat × relbin $\longrightarrow$ conj(nat)

axiomas $(\forall\ x, y, n, m : \text{nat})\ (\forall\ R : \text{relbin})$

$$
\begin{array}{lcl}
\text{rel?}(x, \emptyset, y) & \equiv & \text{false} \\
\text{rel?}(x,\ \text{rel}(n, R, m),\ y) & \equiv & ((x = n) \land (y = m)) \lor \text{rel?}(x, R, y) \\
\text{relacionadosXDer}(x, \emptyset) & \equiv & \emptyset \\
\text{relacionadosXDer}(x,\ \text{rel}(n, R, m)) & \equiv & \textbf{if } (x = n) \textbf{ then} \\
 & & \quad \text{ag}(m,\ \text{relacionadosXDer}(x, R)) \\
 & & \textbf{else} \\
 & & \quad \text{relacionadosXDer}(x, R) \\
 & & \textbf{fi}
\end{array}
$$

Fin TAD

Ejercicio 14

Indique si las siguientes propiedades sobre conjuntos son verdaderas o falsas. Si son verdaderas, realice una demostración. Si son falsas, presente un contraejemplo y pruebe, mediante la aplicación de los axiomas, que la propiedad no se cumple en ese caso.

a) "$(\forall\ A, B : \text{conj}(\alpha))\ (\forall\ n : \alpha)\ (n \in A \cup B \equiv n \in A \lor n \in B)$"
b) "$(\forall\ A, B : \text{conj}(\alpha))\ (\#(A \cup B) = \#(A) + \#(B) \equiv \text{true})$"
c) "$(\forall\ C : \text{conj}(\alpha))\ (\forall\ n : \alpha)\ (n \notin C\ \Rightarrow\ (C - \{n\} \equiv C))$"
d) "$(\forall\ C : \text{conj}(\alpha))\ (\forall\ n : \alpha)\ (\#(\text{Ag}(n, C)) \equiv 1 + \#(C))$"
e) "$(\forall\ C : \text{conj}(\alpha))\ (\forall\ n : \alpha)\ (n \in C - \{n\} \equiv \text{false})$"

Ejercicio 15

Demuestre la siguiente propiedad sobre diccionarios:
"$(\forall\ D : \text{dicc(clave, sign)})\ (\forall\ c : \text{clave})\ ((\text{def?}(c,\ D) \equiv \text{true})\ \Rightarrow\ (\text{def?}(c,\ \text{borrar}(c,\ D)) \equiv \text{false}))$"

Ejercicio 16 ★

Se extiende el tipo Polinomio de la práctica 1 con la siguiente operación (el resto queda igual):

TAD Polinomio extendido

extiende Polinomio

... ...

otras operaciones

derivado : polinomio $\longrightarrow$ polinomio

axiomas $(\forall\ p, q : \text{polinomio})\ (\forall\ n : \text{nat})$

$$
\begin{array}{lcl}
\text{derivado}(\text{cte}(n)) & \equiv & \text{cte}(0) \\
\text{derivado}(\text{X}) & \equiv & \text{cte}(1)
\end{array}
$$

derivado($p + q$)	$\equiv$	derivado(p) + derivado(q)
derivado($p * q$)	$\equiv$	(derivado(p) $*$ q) + (derivado(q) $*$ p)

Fin TAD

Demuestre que
"($\forall$ p : polinomio) (((evaluar(p, 0) = 0?) $\equiv$ true) $\Rightarrow$ (evaluar(p, 1) $\leq$ evaluar(derivado(p), 1) $\equiv$ true))".

Ayuda: Puede utilizar las siguientes propiedades:

1. "($\forall$ x, y : nat) ((($x + y = 0?$) $\equiv$ true) $\Rightarrow$ (($x = 0?$) $\equiv$ true $\wedge$ ($y = 0?$) $\equiv$ true))".
2. "($\forall$ x, y : nat) ((($x \times y = 0?$) $\equiv$ true) $\Rightarrow$ (($x = 0?$) $\equiv$ true $\vee$ ($y = 0?$) $\equiv$ true))".
3. "($\forall$ x, y : nat) (($x \leq y$ $\equiv$ true) $\Rightarrow$ ($\forall$ z : nat) (($x + z \leq y + z$) $\equiv$ true))".

Pruebe al menos una de ellas.

Ejercicio 17

Dado el siguiente tipo abstracto de datos:

TAD Árbol Ternario(α)

géneros at(α)
... ...

observadores básicos

nil?	: at(α)	$\longrightarrow$	bool	
raíz	: at(α) a	$\longrightarrow$	α	$\{(\neg\ \text{nil?}(a))\}$
izq	: at(α) a	$\longrightarrow$	at(α)	$\{(\neg\ \text{nil?}(a))\}$
med	: at(α) a	$\longrightarrow$	at(α)	$\{(\neg\ \text{nil?}(a))\}$
der	: at(α) a	$\longrightarrow$	at(α)	$\{(\neg\ \text{nil?}(a))\}$

generadores

nil	:	$\longrightarrow$	at(α)
tern	: α $\times$ at(α) $\times$ at(α) $\times$ at(α)	$\longrightarrow$	at(α)

otras operaciones

tam	: at(α)	$\longrightarrow$	nat
h	: at(α)	$\longrightarrow$	nat

axiomas ($\forall$ e : α) ($\forall$ t_1, t_2, t_3 : at(α))

tam(nil)	$\equiv$	0
tam(tern(e, t_1, t_2, t_3))	$\equiv$	1 + tam(t_1) + tam(t_2) + tam(t_3)
h(nil)	$\equiv$	0
h(tern(e, t_1, t_2, t_3))	$\equiv$	1 + máx$_3$(h(t_1), h(t_2), h(t_3))

...

Fin TAD

Demuestre que "($\forall$ t : tern(α)) (tam(t) $\leq \frac{3^{\text{h}(t)}-1}{2}$ $\equiv$ true)".

Ayuda: Puede asumir la propiedad "($\forall$ x, y, z : nat) ($3^x + 3^y + 3^z \leq 3^{\text{máx}(x,y,z)+1}$ $\equiv$ true)".

Nota: Sólo se presentan los axiomas más relevantes a efectos del ejercicio.

Ejercicio 18

Dado el siguiente TAD de los procesos secuenciales con *branching*, demuestre que
"($\forall$ p : proc) (trazas(p) $\equiv$ seguir(<>, p))".

TAD PROC

géneros proc

generadores

$0 \quad : \quad \longrightarrow \text{proc}$

$\bullet \bullet \bullet \quad : \text{accion} \times \text{proc} \longrightarrow \text{proc}$

$\bullet + \bullet \quad : \text{proc} \times \text{proc} \longrightarrow \text{proc}$

observadores básicos

$\text{trazas} : \text{proc} \longrightarrow \text{conj(secu(accion))}$

otras operaciones

$\text{seguir} : \text{secu(accion)} \times \text{proc} \longrightarrow \text{conj(secu(accion))}$

$\text{pref} : \text{accion} \times \text{conj(secu(accion))} \longrightarrow \text{conj(secu(accion))}$

axiomas $(\forall\ a : \text{accion})\ (\forall\ p, q : \text{proc})\ (\forall\ w : \text{secu(accion)})\ (\forall\ C : \text{conj(secu(accion))})$

$\text{trazas}(0) \equiv \text{Ag}(<>, \emptyset)$

$\text{trazas}(a \bullet p) \equiv \text{pref}(a, \text{trazas(p)})$

$\text{trazas}(p + q) \equiv \text{trazas(p)} \cup \text{trazas(q)}$

$\text{seguir}(w, 0) \equiv \text{Ag}(w, \emptyset)$

$\text{seguir}(w, a \bullet p) \equiv \text{seguir}(w \circ a, p)$

$\text{seguir}(w, p + q) \equiv \text{seguir}(w, p) \cup \text{seguir}(w, q)$

$\text{pref}(a, c) \equiv \textbf{if } \text{vacio?}(c) \textbf{ then } \emptyset \textbf{ else } \text{Ag}(a \bullet \text{dameUno}(c), \text{pref}(a, \text{sinUno}(c))) \textbf{ fi}$

Fin TAD

TAD ACCIÓN es STRING

Debe demostrar todo lema auxiliar que utilice e involucre al tipo proc.

Puede asumir, sin demostración, el siguiente lema:

"$(\forall\ a : \text{accion})\ (\forall\ c, d : \text{conj(secu(accion))})\ (\text{pref}(a, c \cup d) \equiv \text{pref}(a, c) \cup \text{pref}(a, d))$"

Ejercicio 19 ★

Demuestre por inducción estructural que:

$$(\forall s : \text{secu}(\alpha))(\text{Reverso}(\text{Reverso}(s)) =_{\text{obs}} s)$$

Plantee claramente los lemas necesarios y demostrarlos antes de usarlos en la demostración principal.

Ejercicio 20

Escriba los esquemas de inducción para todos los TADs del apunte de TADs básicos que tengan al menos un generador base y uno recursivo.