Apunte de Inducción Estructural

Algoritmos y Estructuras de Datos 2

1. Introducción

El siguiente es un apunte donde se presenta la resolución de 4 ejercicios de inducción estructural, de los cuales el primero se resolvió en clase.

Si bien este apunte se hizo con buena fe, podría contener errores; por esta razón pedimos que en caso de encontrar alguna falla, o algo que no cierre, nos lo hagan saber para corregirlo.

2. Ejercicio 1: Duplicar una secuencia

2.1. Enunciado

Considerando la siguiente especificación:

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)

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

2.2. Resolución

2.2.1. Convencernos de que lo que queremos probar es cierto

Veamos un ejemplito para ver cómo funciona la función duplicar:

Duplicar(1 $\bullet$ 2 $\bullet$ <>)	por d_2
1 $\bullet$ 1 $\bullet$ Duplicar(2 $\bullet$ <>)	nuevamente usando d_2
1 $\bullet$ 1 $\bullet$ 2 $\bullet$ 2 $\bullet$ Duplicar(<>)	y ahora por d_1
1 $\bullet$ 1 $\bullet$ 2 $\bullet$ 2 $\bullet$ <>	

En este caso vemos que la propiedad parece tener sentido, ya que efectivamente la longitud de la secuencia resultante es el doble de la original.

OJO! esto es sólo un ejemplo para tratar de convencernos. No probamos nada, pero está bueno creernos que lo que queremos demostrar es verdadero.

2.2.2. Plantear la propiedad como predicado unario

El predicado unario es lo que queremos demostrar que vale. Para plantearlo, tenemos que quitar el cuantificador que liga a la variable sobre la que vamos a hacer inducción.

En este caso teníamos:

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

De modo que el predicado es:

$$P(s) = [\ \text{Long}(\text{Duplicar}(s)) \equiv 2 * \text{Long}(s)\ ]$$

Notar que con esta definición de P, la expresión $(\forall s\text{: secu}(\alpha))\ P(s)$ resulta equivalente a lo que queremos demostrar. Ahora podemos usar el esquema como lo aprendimos.

2.2.3. Plantear el esquema de inducción

El esquema de inducción consiste en los casos base y los pasos inductivos que vamos a tener que probar. Este esquema es propio del tipo, ya que se deriva de su conjunto de generadores; los casos base se plantean sobre los generadores básicos y los pasos inductivos sobre los recursivos. Para cualquier propiedad que quieran probar sobre un TAD dado, el esquema de inducción es el mismo.

En este caso tenemos un único generador básico (<>), es decir, que no toma instancias de secuencias, y un único recursivo ($\bullet$), que toma una instancia de secuencia.

Por lo tanto el único caso base es $P(<>)$, y el único paso inductivo es:

$$(\forall s : \text{secu}(\alpha))(\underbrace{P(s)}_{HI} \Rightarrow \underbrace{(\forall a : \alpha)P(a \bullet s)}_{TI})$$

En el paso inductivo $P(s)$ es la **hipótesis inductiva** y $P(a \bullet s)$ es la **tesis inductiva**.

El esquema es entonces:

$$P(<>) \wedge (\forall s : \text{secu}(\alpha))(P(s) \Rightarrow (\forall a : \alpha)P(a \cdot s))$$

Si demostramos esto último, por el principio de inducción estructural habremos demostrado $(\forall s\text{: secu}(\alpha))\ P(s)$.

2.2.4. Probar el caso base

Nuestro caso base era $P(<>)$. Usando la definición que dimos de P, tenemos:

$$P(<>) = [\ \text{Long}(\text{Duplicar}(<>)) \equiv 2 * \text{Long}(<>)\ ]$$

Entonces tenemos que probar:

$$\text{Long}(\text{Duplicar}(<>)) \equiv 2 * \text{Long}(<>)$$

La idea es ir viendo que axiomas podemos ir aplicando para ir reduciendo la expresión que tenemos. Notar que los axiomas nos brindan equivalencias, por lo cual son "reversibles" es decir, se pueden aplicar de izquierda a derecha o de derecha a izquierda.

Usando el axioma l_1, reemplazamos Long(<>) por 0,

$$\text{Long}(\text{Duplicar}(<>)) \equiv 2 * 0$$

Sabemos que 2*0 da 0, entonces podemos reemplazarlo. En general los conocimientos de aritmética sobre los naturales se pueden utilizar sin demostrar, salvo que justamente el ejercicio sea demostrarlo.

$$\text{Long}(\text{Duplicar}(<>)) \equiv 0$$

Mirando el lado izquierdo tenemos Long(Duplicar(<>)). Ninguno de los axiomas de Long aplican, ya que si bien sabemos que Duplicar(<>) es una secuencia, no sabemos como se expresa en función de sus generadores, que es como está axiomatizada Long. Entonces buscamos algún axioma de Duplicar que nos permita seguir. Podemos aplicar d_1.

$$\text{Long}(<>) \equiv 0$$

Ahora sí podemos aplicar un axioma de Long, l_1.

$$0 \equiv 0$$

Y efectivamente 0 es equivalente a 0, por lo cual probamos el caso base.

Hay que tener cuidado de que todos los pasos de la deducción hayan sido "si y sólo si", ya que formalmente la deducción lógica es la que resulta de leer las expresiones desde abajo hacia arriba (partiendo de que sabemos $0 \equiv 0$, deducimos lo que queríamos probar).

2.2.5. Probar el paso inductivo

Nuestro paso inductivo era:

$$(\forall s : \text{secu}(\alpha))(P(s) \Rightarrow (\forall a : \alpha)P(a \bullet s))$$

$P(s)$ es la hipótesis inductiva, a la que vamos a asumir verdadera. Si es falsa, como el paso inductivo es una implicación con $P(s)$ como antecedente, es trivialmente verdadero.

En el paso inductivo aparece una variable a, de tipo α. Es importante notar que la variable está ligada a un cuantificador universal, por lo tanto no se puede asumir nada de ese a, es decir, no se puede suponer que sea un natural mayor que 5, una secuencia de string, nada; sólo sabemos que es de tipo α y nada más.

Entonces, suponiendo que $P(s) = True$, probemos $P(a \bullet s)$ para un a cualquiera.

$$P(a \bullet s) = [\ \text{Long}(\text{Duplicar}(a \bullet s)) \equiv 2 * \text{Long}(a \bullet s)\ ]$$

$$\text{Long}(\text{Duplicar}(a \bullet s)) \equiv 2 * \text{Long}(a \bullet s)$$

por d_2

$$\text{Long}(a \bullet a \bullet \text{Duplicar}(s)) \equiv 2 * \text{Long}(a \bullet s)$$

por l_2

$$1 + \text{Long}(a \bullet \text{Duplicar}(s)) \equiv 2 * \text{Long}(a \bullet s)$$

por l_2 de nuevo

$$1 + 1 + \text{Long}(\text{Duplicar}(s)) \equiv 2 * \text{Long}(a \bullet s)$$

$$2 + \text{Long}(\text{Duplicar}(s)) \equiv 2 * \text{Long}(a \bullet s)$$

aplicando l_2 en el lado derecho

$$2 + \text{Long}(\text{Duplicar}(s)) \equiv 2 * 1 + \text{Long}(s)$$

por propiedad distributiva de los naturales

$$2 + \text{Long}(\text{Duplicar}(s)) \equiv 2 + 2 * \text{Long}(s)$$

Usando la HI, sabemos que $\text{Long}(\text{Duplicar}(s)) \equiv 2 * \text{Long}(s)$

$$2 + 2 * \text{Long}(s) \equiv 2 + 2 * \text{Long}(s)$$

Aquí, como ambos lados son sintácticamente iguales, termina la demostración. Con esto probamos el paso inductivo.

Luego, como probamos antes el caso base, sabemos que la propiedad es válida para cualquier secuencia

3. Ejercicio 2: Tomar y sacar

3.1. Enunciado

Considerando la siguiente especificación:

$\bullet \,\&\, \bullet$: $\text{secu}(\alpha) \times \text{secu}(\alpha) \longrightarrow \text{secu}(\alpha)$

$\&_1)$ $<> \,\&\, t \equiv t$

$\&_2)$ $(a \bullet s) \,\&\, t \equiv \text{a} \bullet (s \,\&\, t)$

Tomar : $\text{secu}(\alpha) \times \text{nat} \longrightarrow \text{secu}(\alpha)$

$t_1)$ $\text{Tomar}(<>, n) \equiv <>$

$t_2)$ $\text{Tomar}(a \bullet s, n) \equiv \textbf{if } n = 0 \textbf{ then } <> \textbf{ else } \text{a} \bullet \text{Tomar}(s, n-1) \textbf{ fi}$

Sacar : $\text{secu}(\alpha) \times \text{nat} \longrightarrow \text{secu}(\alpha)$

$s_1)$ $\text{Sacar}(<>, n) \equiv <>$

$s_2)$ $\text{Sacar}(\text{a} \bullet \text{s}, n) \equiv \textbf{if } n = 0 \textbf{ then } \text{a} \bullet \text{s} \textbf{ else } \text{Sacar}(s, n-1) \textbf{ fi}$

demuestre que:

$$\forall s : \text{secu}(\alpha))(\forall n : \text{nat})(\text{Tomar}(s, n) \,\&\, \text{Sacar}(s, n) \equiv s)$$

3.2. Resolución

3.2.1. Convencernos de que lo que queremos probar es cierto

No lo haremos en el apunte. Hagan los casos de ejemplo que consideren necesarios.

3.2.2. Plantear la propiedad como predicado unario

$$P(s) = (\forall n : \text{nat})(\text{Tomar}(s, n)\ \&\ \text{Sacar}(s, n) \equiv s)$$

3.2.3. Plantear el esquema de inducción

Es el mismo que para el ejercicio anterior:

$$P(<>) \wedge (\forall s : \text{secu}(\alpha))(P(s)\ \Rightarrow\ (\forall a : \alpha)P(a \bullet s))$$

3.2.4. Probar el caso base

$$P(<>)$$
$$(\forall n : \text{nat})(\text{Tomar}(<>, n)\ \&\ \text{Sacar}(<>, n) \equiv <>)$$

sacamos el cuantificador n, pero teniendo en cuenta que no podemos suponer nada de n, que es un nat arbitrario

$$\text{Tomar}(<>, n)\ \ \&\ \ \text{Sacar}(<>, n) \equiv <>$$

usamos t_1

$$<>\ \ \&\ Sacar(<>, n) \equiv <>$$

usamos s_1

$$<>\ \ \&\ \ <> \equiv <>$$

usamos $\&_1$

$$<> \equiv <>$$

3.2.5. Probar el paso inductivo

$$P(a \bullet s)$$
$$(\forall n : nat)(\text{Tomar}(a \bullet s, n)\ \&\ \text{Sacar}(a \bullet s, n) \equiv a \bullet s)$$
$$\text{Tomar}(a \bullet s, n)\ \&\ \text{Sacar}(a \bullet s, n) \equiv a \bullet s$$

por t_2

$$(\textbf{if } n = 0 \textbf{ then } <>\ \textbf{ else } a \bullet \text{Tomar}(s, n-1)\ \textbf{fi})\ \&\ \text{Sacar}(a \bullet s, n) \equiv a \bullet s$$

podemos meter el & dentro de cada rama del if (ver Anexo):

$$\textbf{if } n = 0 \textbf{ then } (<>\ \&\ \text{Sacar}(a \bullet s, n))\ \textbf{else } (a \bullet \text{Tomar}(s, n-1))\ \&\ \text{Sacar}(a \bullet s, n)\ \textbf{fi} \equiv a \bullet s$$

aplicando s_2

a • s ≡ **if** n = 0 **then**
(<> & **if** n = 0 **then** (a • s) **else** Sacar(s, n-1) **fi**)
else
((a • Tomar(s, n - 1)) & **if** n = 0 **then** <> **else** Sacar(s, n-1) **fi**)
fi

aquí podemos ver que en la rama del **then** en el **if** principal, no tiene sentido preguntar nuevamente si $n = 0$, ya que si llegamos ahí era porque efectivamente valía la guarda, por lo cual, nos podemos quedar sólo con la parte **then** en el **if** interno (ver propiedad en el Anexo)

a • s ≡ **if** n = 0 **then**
(<> & (a • s))
else
((a • Tomar(s, n - 1)) & **if** n = 0 **then** <> **else** Sacar(s, n-1) **fi**)
fi

por $\&_1$:

a • s ≡ **if** n = 0 **then**
(a • s)
else
((a • Tomar(s, n - 1)) & **if** n = 0 **then** <> **else** Sacar(s, n-1) **fi**)
fi

al contrario de lo que pasaba antes, ahora miramos la rama del **else**, y por lo tanto sabemos que vale $n \neq 0$, por lo cual del segundo **if** nos podemos quedar sólo con la parte del **else** (esto también se obtiene componiendo propiedades que aparecen en el Anexo):

a • s ≡ **if** n = 0 **then** (a • s) **else** ((a • Tomar(s, n - 1)) & Sacar(s, n-1)) **fi**

aplicando ahora $\&_2$

a • s ≡ **if** n = 0 **then** (a • s) **else** (a • (Tomar(s, n - 1) & Sacar(s, n-1))) **fi**

recordemos que la hipótesis inductiva es $P(s)$, y por lo tanto vale para todo posible segundo parámetro, inclusive $n - 1$,

a • s ≡ **if** n = 0 **then** (a • s) **else** (a • s) **fi**

como ambas ramas del **if** desembocan en lo mismo, tenemos una vez más una propiedad que nos permite reducir simplemente

a • s ≡ a • s

Es importante notar de este ejercicio lo útil que es tener las propiedades del **if** probadas previamente, ya que si no habría que hacer toda la consideración de casos dentro de esta demostración, ensuciando la misma.

4. Ejercicio 3: Diccionarios múltiples

4.1. Enunciado

Considerando la siguiente especificación:

TAD DiccMul

observadores básicos

sigsActuales : clave × diccmul(clave × sig) $\longrightarrow$ conj(sig)

sigsArcaicos : clave × diccmul(clave × sig) $\longrightarrow$ conj(sig)

generadores

vacio : $\longrightarrow$ diccmul(clave, sig)

definir : clave c × sig s × diccmul(clave × sig) d $\longrightarrow$ diccmul(clave, sig) {s $\notin$ sigsArcaicos(c, d)}

enDesuso : clave c × sig s × diccmul(clave × sig) d $\longrightarrow$ diccmul(clave, sig) {s $\in$ sigsActuales(c, d)}

axiomas $\forall c_1, c_2 : clave, \forall s : sig, \forall d : diccmul(clave, sig)$

$s1$) sigsActuales(c_1, vacio) $\equiv \emptyset$

$s2$) sigsActuales(c_1, definir(c_2, s, d)) $\equiv$
if $c_1 =_{clave} c_2$ **then**
Ag(s, sigsActuales(c_1, d))
else
sigsActuales(c_1, d)
fi

$s3$) sigsActuales(c_1, enDesuso(c_2, s, d)) $\equiv$
if $c_1 =_{clave} c_2$ **then**
sigsActuales(c_1, d)-{s}
else
sigsActuales(c_1, d)
fi

$sa1$) sigsArcaicos($c_1, vacio$) $\equiv \emptyset$

$sa2$) sigsArcaicos(c_1, definir(c_2, s, d)) $\equiv$
sigsArcaicos(c_1, d)

$sa2$) sigsArcaicos(c_1, enDesuso(c_2, s, d)) $\equiv$
if $c_1 =_{clave} c_2$ **then**
Ag(s, sigsArcaicos(c_1, d))
else
sigsArcaicos(c_1, d)
fi

Fin TAD

Quremos demostrar:

$\forall c : clave, \forall d : diccmul(clave, sig)\ (sigsActuales(c, d) \cap sigsArcaicos(c, d) \equiv \emptyset)$

4.1.1. Convencernos de que lo que queremos probar es cierto

Tarea :)

4.1.2. Plantear la propiedad como predicado unario

Tenemos:

$\forall c : clave, \forall d : diccmul(clave, sig)\ (sigsActuales(c, d) \cap sigsArcaicos(c, d) \equiv \emptyset)$

De modo que el predicado resultante es:

$P(d) = \forall c : clave\ (sigsActuales(c, d) \cap sigsArcaicos(c, d) \equiv \emptyset)$

4.1.3. Plantear el esquema de inducción

Tenemos un constructor base y dos constructores recursivos. Notemos que los generadores recursivos tienen restricciones, que hay que tener en cuenta. Es decir, cuando aplicamos definir no podemos dar cualquier significado, sino que tiene que ser un significado que no este en los arcaicos. Algo similar ocurre para enDesuso.

$$P(vacio) \wedge$$
$$\forall d : diccmul(clave, sig)(\underbrace{P(d)}_{HI} \Rightarrow \underbrace{(\forall c_1 : clave, s : sig(s \notin sigsArcaicos(c_1, d) \Rightarrow_L P(definir(c_1, s, d))))}_{TI}) \wedge$$
$$\forall d : diccmul(clave, sig)(\underbrace{P(d)}_{HI} \Rightarrow \underbrace{(\forall c_1 : clave, s : sig(s \in sigsActuales(c_1, d) \Rightarrow_L P(enDesuso(c_1, s, d))))}_{TI})$$

4.1.4. Probar el caso base

$$P(vacio)$$
$$\forall c : clave\ sigsActuales(c, vacio) \cap sigsArcaicos(c, vacio) \equiv \emptyset$$

Saco el cuantificador, pero no lo olvidamos, es decir no vamos a pedir nada sobre c!

$$sigsActuales(c, vacio) \cap sigsArcaicos(c, vacio) \equiv \emptyset$$

por s1

$$\emptyset \cap sigsArcaicos(c, vacio) \equiv \emptyset$$

por sa1

$$\emptyset \cap \emptyset \equiv \emptyset$$

por el axioma de intersección

$$\emptyset \equiv \emptyset$$

De esta forma probamos el caso base

4.1.5. Probar los pasos inductivos

Probemos primero el paso inductivo de definir.

Tenemos:

$P(d) \Rightarrow (\forall c_1 : clave, s : sig(s \notin sigsArcaicos(c_1, d) \Rightarrow P(definir(c_1, s, d))))$

P(d) la vamos a suponer como verdadera, es nuestra hipótesis inductiva, nos queda ver entonces que

$(\forall c_1 : clave, s : sig(s \notin sigsArcaicos(c_1, d) \Rightarrow_L P(definir(c_1, s, d))))$

Por simplicidad saco los cuantificadores: $s \notin sigsArcaicos(c_1, d) \Rightarrow_L P(definir(c_1, s, d))$

Si no se cumple el antecedente, y s está en los significados arcaicos, no hay nada que probar. Por eso vamos a considerar el caso donde vale $s \notin sigsArcaicos(c_1, d)$

Expandiendo $P(definir(c_1, s, d))$

$$\forall c : clave\ sigsActuales(c, definir(c_1, s, d)) \cap sigsArcaicos(c, definir(c_1, s, d)) \equiv \emptyset$$

Saco el cuantificador por comodidad

$$sigsActuales(c, definir(c_1, s, d)) \cap sigsArcaicos(c, definir(c_1, s, d)) \equiv \emptyset$$

Por s2

$$sigsActuales(c, definir(c_1, s, d)) \cap sigsArcaicos(c, definir(c_1, s, d)) \equiv \emptyset$$

$$(if\ c =_{clave} c_1\ then\ Ag(s, sigsActuales(c, d))\ else\ sigsActuales(c, d)) \cap sigsArcaicos(c, definir(c_1, s, d)) \equiv \emptyset$$

por sa2

$$(if\ c =_{clave} c_1\ then\ Ag(s, sigsActuales(c, d))\ else\ sigsActuales(c, d)) \cap sigsArcaicos(c, d) \equiv \emptyset$$

Bueno ahora vamos a separar en casos, o bien $c =_{clave} c_1$ o bien son distintas. Es muy importante prestar atención cuando se separa en casos, hay que recordar que caso se está considerando y asegurarse que se consideren todos, es decir que no nos olvidemos de probar ningún caso. Si hay un if sólo, esto es relativamente fácil, pero cuando empiezan a aparecer mas y la cantidad de casos aumenta hay que ser mas cuidadoso.

Supongamos primero que son distintos, entonces entramos por la rama del else:

$$(if\ c =_{clave} c_1\ then\ Ag(s, sigsActuales(c, d))\ else\ sigsActuales(c, d)) \cap sigsArcaicos(c, d) \equiv \emptyset$$

$$sigsActuales(c, d) \cap sigsArcaicos(c, d) \equiv \emptyset$$

Y esto sabemos que es verdadero, por hipótesis inductiva!

¿Qué pasa si eran iguales las claves? Entramos por la rama del then:

$$(if\ c =_{clave} c_1\ then\ Ag(s, sigsActuales(c, d))\ else\ sigsActuales(c, d)) \cap sigsArcaicos(c, d) \equiv \emptyset$$

$$Ag(s, sigsActuales(c, d)) \cap sigsArcaicos(c, d) \equiv \emptyset$$

Podemos aplicar el axioma de la intersección:

$$(if\ s \in sigsArcaicos(c, d)\ thenAg(s, sigsActuales(c, d) \cap sigsArcaicos(c, d))$$

$$else\ sigsActuales(c, d) \cap sigsArcaicos(c, d)) \equiv \emptyset$$

Para que se cumpla la restricción de definir, tiene que valer $s \notin sigsArcaicos(c_1, d)$ (lo vimos antes) y como $c =_{clave} c_1$, vale $s \notin sigsArcaicos(c, d)$. Por esta razón, entramos por la rama del else:

$(if\ s \in sigsArcaicos(c,d)\ then\ Ag(s, sigsActuales(c,d) \cap sigsArcaicos(c,d))\ else\ sigsActuales(c,d) \cap sigsArcaicos(c,d)) \equiv \emptyset$

$$sigsActuales(c,d) \cap sigsArcaicos(c,d) \equiv \emptyset$$

Y nuevamente, esto es cierto por HI

Probemos ahora el paso inductivo para enDesuso:

$P(d) \Rightarrow (\forall c_1 : clave, s : sig(s \in sigsActuales(c_1, d) \Rightarrow_L P(enDesuso(c_1, s, d)))$

Nuevamente P(d) va a ser nuestra hipótesis inductiva, y nuevamente también tenemos una restricción sobre s. Vamos a suponer que la restricción sobre s vale, porque si no, no tenemos nada que probar.

Concentrémonos entonces en $P(enDesuso(c_1, s, d)))$

$$P(enDesuso(c_1, s, d))$$
$$\forall c : clave\ sigsActuales(c, enDesuso(c_1, s, d)) \cap sigsArcaicos(c, enDesuso(c_1, s, d)) \equiv \emptyset$$
$$sigsActuales(c, enDesuso(c_1, s, d)) \cap sigsArcaicos(c, enDesuso(c_1, s, d)) \equiv \emptyset$$

por s3

$$(if\ c =_{clave} c_1\ then\ sigsActuales(c,d) - \{s\}\ else\ sigsActuales(c,d)) \cap sigsArcaicos(c, enDesuso(c_1, s, d)) \equiv \emptyset$$

Otra vez podemos separar en casos, según si $c =_{clave} c_1$ o si no.

Si consideramos que $c \neq_{clave} c_1$, entramos por la rama del else:

$$(if\ c =_{clave} c_1\ then\ sigsActuales(c,d) - \{s\}\ else\ sigsActuales(c,d)) \cap sigsArcaicos(c, enDesuso(c_1, s, d)) \equiv \emptyset$$

$$sigsActuales(c,d) \cap sigsArcaicos(c, enDesuso(c_1, s, d)) \equiv \emptyset$$

Aplicamos sa3

$$sigsActuales(c,d) \cap (if\ c =_{clave} c_1\ then\ Ag(s, sigsArcaicos(c,d))\ else\ sigsArcaicos(c,d)) \equiv \emptyset$$

Recordemos que estamos suponiendo que c es distinta a c_1, por lo tanto acá también nos vamos a la rama del else:

$$sigsActuales(c,d) \cap (if\ c =_{clave} c_1\ then\ Ag(s, sigsArcaicos(c,d))\ else\ sigsArcaicos(c,d)) \equiv \emptyset$$

$$sigsActuales(c,d) \cap sigsArcaicos(c,d) \equiv \emptyset$$

Y esto es cierto, por hipótesis inductiva

Ahora hagamos el caso en el que vale que $c =_{clave} c_1$:

En este caso entramos por la rama del then:

$$(if\ c =_{clave} c_1\ then\ sigsActuales(c,d) - \{s\}\ else\ sigsActuales(c,d)) \cap sigsArcaicos(c, enDesuso(c_1, s, d)) \equiv \emptyset$$

$$sigsActuales(c,d) - \{s\} \cap sigsArcaicos(c, enDesuso(c_1, s, d)) \equiv \emptyset$$

por sa3:

$$sigsActuales(c,d) - \{s\} \cap (if\ c =_{clave} c_1\ then\ Ag(s, sigsArcaicos(c,d))\ else\ sigsArcaicos(c,d)) \equiv \emptyset$$

Como suponemos que $c =_{clave} c_1$:

$$sigsActuales(c,d) - \{s\} \cap Ag(s, sigsArcaicos(c,d)) \equiv \emptyset$$

No tenemos ningún axioma para seguir, entonces hay que pensar. Mirando la expresión que nos quedo vemos que tiene sentido: Sabemos que $sigsActuales(c, d) \cap sigsArcaicos(c, d) \equiv \emptyset$ por HI, ademas s está en $sigsActuales(c, d)$, ya que $c =_{clave} c_1$ y $s \in sigsActuales(c_1, d)$ era una restricción que le pedimos a s. Si bien s se agrega a $sigsArcaicos(c, d)$, se la quita de $sigsActuales(c, d)$ por lo cual la intersección no debería cambiar.

Lo que vamos a hacer es plantear un lema, o sea un resultado auxiliar que nos permita continuar con nuestra demostración. Luego probaremos que el lema es cierto, si no lo fuera nuestra demostración no tendría validez.

El lema que queremos es:

$$(\forall a, b : conj(\alpha), \forall e : \alpha)(a \cap b \equiv \emptyset \Rightarrow (a - \{e\} \cap Ag(e, b) \equiv \emptyset))$$

Si aplicamos el lema, ya tenemos probado el segundo paso inductivo.

Ahora hay que probar el lema. Los pasos son los mismos, y ya nos convencimos de que anda, entonces pasemos a plantear el predicado unario. Hasta ahora siempre las propiedades tenían una única variable del tipo sobre el que hacíamos inducción. En este lema hay dos variables de tipo conjunto. ¿Sobre cuál trabajamos?

La respuesta a esa pregunta depende del conjunto de axiomas que tengamos, en este caso los de conjunto, suelen hacer inducción sobre el primer parámetro, por lo cual nos conviene plantear el predicado unario como función de a y no como función de b(Tarea: probar plantearlo como P(b)).

De acuerdo con esto, el predicado es:

$$P(a) = (\forall b : conj(\alpha), \forall e : \alpha)(a \cap b \equiv \emptyset \Rightarrow (a - \{e\} \cap Ag(e, b) \equiv \emptyset))$$

Planteamos el esquema de inducción:

$$P(\emptyset) \wedge (\forall e_1 : \alpha)(P(a) \Rightarrow P(Ag(e, a)))$$

El caso base:

$$P(\emptyset)$$

$$(\forall b : conj(\alpha), \forall e : \alpha)(\emptyset \cap b \equiv \emptyset \Rightarrow (\emptyset - \{e\} \cap Ag(e, b) \equiv \emptyset))$$

El predicado es una implicación. Veamos que el antecedente es verdadero:

$$\emptyset \cap b \equiv \emptyset$$

Por el primer axioma de $\cap$

$$\emptyset \equiv \emptyset$$

Concentrémonos entonces en el consecuente:

$$\emptyset - \{e\} \cap Ag(e, b) \equiv \emptyset$$

Por el primer axioma de $-\{\cdot\}$

$$\emptyset \cap Ag(e, b) \equiv \emptyset$$

Por el primer axioma de $\cap$

$$\emptyset \equiv \emptyset$$

Ya tenemos el caso base.

Probemos ahora el paso inductivo

$$\forall e : \alpha(P(a) \Rightarrow P(Ag(e, a))$$

$$P(Ag(e, a))$$

$$Ag(e_1, a) \cap b \equiv \emptyset \Rightarrow (Ag(e_1, a) - \{e\} \cap Ag(e, b) \equiv \emptyset))$$

Estudiemos el antecedente, recordemos que nos va a interesar sólo el caso en el que es verdadero:

$$Ag(e_1, a) \cap b \equiv \emptyset$$

Por el axioma 2 de $\cap$

$$if\ e_1 \in b\ then\ Ag(e_1, a \cap b)\ else\ a \cap b \equiv \emptyset$$

Si la guarda del if es verdadera, nunca puede ocurrir que $Ag(e_1, a) \cap b \equiv \emptyset$, ya que el resultado de la intersección tiene por lo menos un elemento. Entonces la guarda tiene que ser falsa. De acá vemos que tiene que ocurrir que $\neg e_1 \in b$. Por otro lado como la guarda es falsa, vemos que $a \cap b \equiv \emptyset$

Vayamos ahora al consecuente:

$$Ag(e_1, a) - \{e\} \cap Ag(e, b) \equiv \emptyset$$

Por el axioma 2 de $-\{\cdot\}$

$$(if\ e_1 = e\ then\ a - \{e\}\ else\ Ag(e_1, a - \{e\})) \cap Ag(e, b) \equiv \emptyset$$

Separemos en casos:

Caso 1: $e_1 = e$

Entramos por la rama del then:

$$(if\ e_1 = e\ then\ a - \{e\}\ else\ Ag(e_1, a - \{e\})) \cap Ag(e, b) \equiv \emptyset$$

$$a - \{e\} \cap Ag(e, b) \equiv \emptyset$$

Por lo que vimos en el antecedente, $a \cap b \equiv \emptyset$, entonces por HI

$$a - \{e\} \cap Ag(e, b) \equiv \emptyset$$

$$\emptyset \equiv \emptyset$$

Caso 2: $e_1 \neq e$

Vamos a la rama del else:

$$Ag(e_1, a - \{e\}) \cap Ag(e, b) \equiv \emptyset$$

Por el axioma 2 de $\cap$

$$if\ e_1 \in Ag(e, b)\ then\ Ag(e_1, a - \{e\} \cap Ag(e, b))\ else\ a - \{e\} \cap Ag(e, b)) \equiv \emptyset$$

Por el axioma 2 de $\in$

$$if\ e_1 = e \vee e_1 \in b\ then\ Ag(e_1, a - \{e\} \cap Ag(e, b))\ else\ a - \{e\} \cap Ag(e, b)) \equiv \emptyset$$

Ahora bien, estamos suponiendo que $e_1 = e$ no vale, y vimos, cuando analizamos el antecedente, que $e_1 \in b$ es falso. Entonces entramos por la rama del else:

$$a - \{e\} \cap Ag(e, b) \equiv \emptyset$$

Como vimos antes $a \cap b \equiv \emptyset$, aplicamos la HI y terminamos.

Con esto probamos el lema. Al probarlos, terminamos la demostración de la propiedad.

5. Ejercicio 4: Árboles

5.1. Enunciado

Demostrar que para todo árbol binario de números naturales con al menos un nodo, la suma del valor de sus nodos es mayor igual a la suma del valor de sus hojas. En otras palabras, probar

$$(\forall A \in Ab(Nat))\,(\neg Nil?(A) \;\Rightarrow\; (SHojas(A) \leq SNodos(A)))$$

Con

SHojas : Ab(Nat) $\longrightarrow$ Nat

$sH1$) SHojas(Nil) $\equiv$ 0

$sH2$) sHojas(Ab(Izq, N, Der)) $\equiv$ **if** (Nil?(Izq) $\wedge$ Nil?(Der)) **then**
N
else
SHojas(Izq) + SHojas(Der)
fi

SnODOS : Ab(Nat) $\longrightarrow$ Nat

$sn1$) sNodos(Nil) $\equiv$ 0

$sn2$) sNodos(Ab(Izq, N, Der)) $\equiv$ N + SNodos(Izq) + SNodos(Der)

5.2. Resolución

5.2.1. Convencernos de que lo que queremos probar es cierto

La propiedad que nos piden probar dice que si nuestro árbol no es Nil entonces la suma de sus hojas es menor a la de sus nodos. Notemos que todas sus hojas son nodos y sus valores no son negativos, por lo que si sumamos un subconjunto de números no negativos (SHojas), obtenemos un número menor igual que si sumamos todo el conjunto (SNodos).

5.2.2. Plantear la propiedad como predicado unario

$$P(A) = \neg Nil?(A) \;\Rightarrow\; (SHojas(A) \leq SNodos(A))$$

5.2.3. Plantear el esquema de inducción

A diferencia de los ejercicios anteriores, los Árboles Binarios reciben dos instancias del tipo $Ab(\alpha)$, por lo tanto nuestra hipótesis inductiva cambiará:
Caso base:

$$P(Nil)$$

Paso recursivo:

$$(\forall Izq, Der : Ab(Nat))\ \underbrace{(P(Izq) \wedge P(Der))}_{HI} \Rightarrow (\forall N : Nat)\ \underbrace{P(Ab(Izq, N, Der))}_{TI}$$

En este caso, la hipótesis inductiva aplica para dos árboles Izq y Der. El esquema queda entonces:

$$P(Nil) \wedge ((\forall Izq, Der : Ab(Nat))(\underbrace{(P(Izq) \wedge P(Der))}_{HI} \Rightarrow (\forall N : Nat)\ \underbrace{P(Ab(Izq, N, Der))))}_{TI}$$

5.2.4. Probar los casos base

Nuestro caso base consiste en probar que la propiedad vale para Nil, notemos que la propiedad es una implicación, por lo cual hay que prestar atención a que el antecedente sea verdadero, como este no es el caso, el caso base esta probado.

$$\begin{aligned} \neg Nil?(Nil) \ &\Rightarrow\ (SHojas(Nil) \leq SNodos(Nil)) \\ \equiv False \ &\Rightarrow\ (SHojas(Nil) \leq SNodos(Nil)) \\ &\equiv True \end{aligned}$$

5.2.5. Probar los pasos inductivos

Para esta sección asumiremos que tenemos dos árboles Izq y Der tales que $P(Izq) \wedge P(Der) \equiv True$, probaremos entonces que dado cualquier número natural N, P(Ab(Izq, N, Der)) es verdadera.

$$P(Ab(Izq, N, Der)) \equiv (\neg Nil?(Ab(Izq, N, Der)) \ \Rightarrow\ (SHojas(Ab(Izq, N, Der)) \leq SNodos(Ab(Izq, N, Der))))$$

Por definición de Nil?,

$$Nil?(Ab(Izq, N, Der)) \equiv false$$

Luego queremos ver que el otro lado de la implicación es verdadero. O sea,

$$(SHojas(Ab(Izq, N, Der)) \leq SNodos(Ab(Izq, N, Der))) \equiv true$$

Por sH2, la primera parte es equivalente a:

$$\begin{aligned} &\textbf{if } (Nil?(Izq) \wedge Nil?(Der) \textbf{ then } N \\ &\textbf{else } SHojas(Izq) + SHojas(Der) \textbf{ fi }) \\ &\leq SNodos(Ab(Izq, N, Der)) \end{aligned}$$

Usando la propiedad del If para funciones esto equivale a:

$$\begin{aligned} &\textbf{if } (Nil?(Izq) \wedge Nil?(Der)) \\ &\textbf{then } N \leq SNodos(Ab(Izq, N, Der)) \\ &\textbf{else } (SHojas(Izq) + SHojas(Der)) \leq SNodos(Ab(Izq, N, Der)) \textbf{ fi} \end{aligned}$$

Luego, reemplazando $SNodos(Ab(Izq, N, Der))$ por su definición, o sea, por sN2, obtenemos:

$$\textbf{if } (Nil?(Izq) \land Nil?(Der))$$
$$\textbf{then} N \leq (N + SNodos(Izq) + SNodos(Der))$$
$$\textbf{else}(SHojas(Izq) + SHojas(Der)) \leq (N + SNodos(Izq) + SNodos(Der))) \textbf{ fi}$$

Llegamos a una instancia donde lo que queremos ver es que sean como sean Izq y Der el If siempre devuelve True. Separemos en casos:

1. Izq y Der son Nil
2. Izq es Nil y Der no lo es
3. Izq no es Nil, pero Der si lo es
4. Ninguno de los dos es Nil

Caso 1: Si Izq y Der son Nil

$$\textbf{if } (Nil?(Izq) \land Nil?(Der))$$
$$\textbf{then } N \leq (N + SNodos(Izq) + SNodos(Der))$$
$$\textbf{else } (SHojas(Izq) + SHojas(Der)) \leq (N + SNodos(Izq) + SNodos(Der)))$$
$$\textbf{fi}$$

$$\equiv N \leq (N + SNodos(Izq) + SNodos(Der))$$

Tanto Izq como Der son Nil, por sN1,

$$\equiv N \leq N + 0 + 0 \equiv true$$

Por lo tanto, si ambos son Nil el If devuelve true

Caso 2/3: Sin perdida de generalidad asumiremos que Izq es Nil y Der no lo es. Notemos que al ser las propiedades simétricas para ambos árboles hijos este análisis es idéntico para el otro caso.

$$\textbf{if } (Nil?(Izq) \land Nil?(Der))$$
$$\textbf{then } N \leq (N + SNodos(Izq) + SNodos(Der))$$
$$\textbf{else } (SHojas(Izq) + SHojas(Der)) \leq (N + SNodos(Izq) + SNodos(Der)))$$
$$\textbf{fi}$$

Como Der no es Nil la conjunción no es true,

$$\equiv (SHojas(Izq) + SHojas(Der)) \leq N + SNodos(Izq) + SNodos(Der)$$

Por sH0,

$$\equiv (0 + SHojas(Der)) \leq N + SNodos(Izq) + SNodos(Der)$$

Por sN0,

$$\equiv SHojas(Der) \leq N + 0 + SNodos(Der)$$

Como Der no es Nil, por Hipótesis Inductiva $SHojas(Der) \leq SNodos(Der)$ Luego, como $0 \leq N$,

$$\equiv (SHojas(Der) \leq N + SNodos(Der)) \equiv true$$

Por lo tanto, si alguno de los árboles hijos es Nil y el otro no, el If retorna true.

Caso 4: Asumimos que tanto Izq como Der no son Nil, luego,

$$\textbf{if } (Nil?(Izq) \land Nil?(Der))$$
$$\textbf{then } N \leq (N + SNodos(Izq) + SNodos(Der))$$
$$\textbf{else } (SHojas(Izq) + SHojas(Der)) \leq (N + SNodos(Izq) + SNodos(Der)))$$
$$\textbf{fi}$$

$$\equiv (SHojas(Izq) + SHojas(Der)) \leq N + SNodos(Izq) + SNodos(Der)$$

Como Izq no es Nil, por Hipótesis Inductiva,

$$SHojas(Izq) \leq SNodos(Izq)$$

Además, Der no es Nil, por lo tanto, por HI:

$$SHojas(Der) \leq SNodos(Der)$$

Sumando ambas inecuaciones,

$$SHojas(Izq) + SHojas(Der) \leq SNodos(Izq) + SNodos(Der)$$

Como N es mayor igual a 0,

$$SNodos(Izq) + SNodos(Der) \leq N + SNodos(Izq) + SNodos(Der)$$

Por lo tanto,

$$(SHojas(Izq) + SHojas(Der)) \leq SNodos(Izq) + SNodos(Der) \leq N + SNodos(Izq) + SNodos(Der)$$

$$true \equiv SHojas(Izq) + SHojas(Der) \leq N + SNodos(Izq) + SNodos(Der)$$

Esto significa que en el caso de que Der y Izq no son Nil el If retorna true.

Dado que existen solo 4 casos y en todos ellos el If retorna true podemos concluir que

$$P(Ab(Izq, N, Der)) \equiv true$$

Por lo tanto, para todo Árbol Binario de Naturales la propiedad vale como queriamos mostrar.

6. Anexo: Propiedades válidas sobre el if

Estas son algunas de las propiedades relacionadas con **if** que se pueden utilizar sin demostrar. Pueden demostrarlas como ejercicio si quieren, es fácil.

- **if** p **then** q **else** q **fi** $\equiv q$
- función(**if** p **then** q **else** r **fi**) $\equiv$ **if** p **then** función(q) **else** función(r) **fi** [1]
- **if** $\neg p$ **then** q **else** r **fi** $\equiv$ **if** p **then** r **else** q **fi**
- **if** p **then** (**if** p **then** q **else** r **fi**) **else** t **fi** $\equiv$ **if** p **then** q **else** t **fi**

[1] Vale también si función toma mas parámetros