Elementos de matemáticas formalizados en Isabelle/HOL

Carlos Núñez Fernández

18 de noviembre de 2019

Índice

Capítulo 1

Suma de los primeros números impares

1.1 Demostración en lenguaje natural

El primer teorema es una propiedad de los números naturales.

Teorema 1.1.1 *La suma de los n primeros números impares es n^2.*

Demostración: La demostración la haremos en inducción sobre n.

(Base de la inducción) El caso $n = 0$ es trivial.

(Paso de la inducción) Supongamos que la propiedad se verifica para n y veamos que también se verifica para $n + 1$.

Tenemos que demostrar que $\sum_{j=1}^{n+1} k_j = (n + 1)^2$ donde k_j el j–ésimo impar; es decir, $k_j = 2j - 1$.

$$
\begin{aligned}
&\textstyle\sum_{j=1}^{n+1} k_j \\
&= k_{n+1} + \textstyle\sum_{j=1}^{n} k_j \\
&= k_{n+1} + n^2 \\
&= 2(n + 1) - 1 + n^2 \\
&= n^2 + 2n + 1 \\
&= (n + 1)^2
\end{aligned}
$$

□

1.2 Especificación en Isabelle/HOL

Para especificar el teorema en Isabelle, se comienza definiendo la función *suma-impares* tal que *suma-impares n* es la suma de los *n* primeros números impares

fun *suma-impares* :: *nat* $\Rightarrow$ *nat* **where**
suma-impares 0 $=$ *0*
| *suma-impares* $(Suc\ n) = (2{*}(Suc\ n) - 1) +$ *suma-impares n*

El enunciado del teorema es el siguiente:

lemma *suma-impares* $n = n * n$
oops

1.3 Demostración aplicativa

En la demostración se usará la táctica *induct* que hace uso del esquema de inducción sobre los naturales:

$$\frac{P\ 0 \qquad \bigwedge nat.\ \dfrac{P\ nat}{P\ (Suc\ nat)}}{P\ nat} \tag{nat.induct}$$

Vamos a presentar distintas demostraciones del teorema. La primera es la demostración aplicativa detallada.

lemma *suma-impares* $n = n * n$
apply (*induct n*)
apply (*simp only*: *suma-impares.simps*(*1*))
apply (*simp only*: *suma-impares.simps*(*2*))
apply (*simp only*: *mult-Suc mult-Suc-right*)
done

En la demostración anterior hemos usado dentro del método *simp* únicamente la definición de *suma-impares*, para ello lo hemos indicado con *simp only*. A parte hemos usado lo siguiente :

$$Suc\ m * n = n + m * n \tag{mult-Suc}$$

$$m * Suc\ n = m + m * n \tag{mult-Suc-right}$$

Se puede eliminar los detalles de la demostración anterior.

```
lemma suma-impares n = n * n
 apply (induct n)
  apply simp-all
 done
```

1.4 Demostración automática

La correspondiente demostración automática es

```
lemma suma-impares n = n * n
 by (induct n) simp-all
```

1.5 Demostración estructurada

La demostración estructurada y detallada del lema anterior es

```
lemma suma-impares n = n * n
proof (induct n)
 have suma-impares 0 = 0
  by (simp only: suma-impares.simps(1))
 also have ... = 0 * 0
  by (simp only: mult-0)
 finally show suma-impares 0 = 0 * 0
  by simp
next
 fix n
 assume HI: suma-impares n = n * n
 have suma-impares (Suc n) = (2 * (Suc n) - 1) + suma-impares n
  by (simp only: suma-impares.simps(2))
 also have ... = (2 * (Suc n) - 1) + n * n
  by (simp only: HI)
 also have ... = n * n + 2 * n + 1
  by (simp only: mult-Suc-right)
 also have ... = (Suc n) * (Suc n)
  by (simp only: mult-Suc mult-Suc-right)
 finally show suma-impares (Suc n) = (Suc n) * (Suc n)
  by simp
qed
```

En la demostración anterior se pueden ocultar detalles.

lemma *suma-impares n* $= n * n$
proof (*induct n*)
 show *suma-impares 0* $= 0 * 0$ **by** *simp*
next
 fix *n*
 assume *HI*: *suma-impares n* $= n * n$
 have *suma-impares* (*Suc n*) $=$ ($2 *$ (*Suc n*) $- 1$) $+$ *suma-impares n*
  by *simp*
 also have ... $=$ ($2 *$ (*Suc n*) $- 1$) $+ n * n$
  using *HI* **by** *simp*
 also have ... $=$ (*Suc n*) $*$ (*Suc n*)
  by *simp*
 finally show *suma-impares* (*Suc n*) $=$ (*Suc n*) $*$ (*Suc n*)
  by *simp*
qed

1.6 Demostración con patrones

La demostración anterior se puede simplificar usando patrones.

lemma *suma-impares n* $= n * n$ (**is** *?P n* $=$ *?Q n*)
proof (*induct n*)
 show *?P 0* $=$ *?Q 0* **by** *simp*
next
 fix *n*
 assume *HI*: *?P n* $=$ *?Q n*
 have *?P* (*Suc n*) $=$ ($2 *$ (*Suc n*) $- 1$) $+$ *suma-impares n*
  by *simp*
 also have ... $=$ ($2 *$ (*Suc n*) $- 1$) $+ n * n$ **using** *HI* **by** *simp*
 also have ... $=$ *?Q* (*Suc n*) **by** *simp*
 finally show *?P* (*Suc n*) $=$ *?Q* (*Suc n*) **by** *simp*
qed

La demostración usando otro patrón es

lemma *suma-impares n* $= n * n$ (**is** *?P n*)
proof (*induct n*)
 show *?P 0* **by** *simp*
next
 fix *n*

assume *?P n*
then show *?P* (*Suc n*) **by** *simp*
qed

Capítulo 2

Cancelación de funciones inyectivas

2.1 Demostración en lenguaje natural

Comentario 1: Estructurar en secciones.

Comentario 2: Hacer demostraciones detalladas.

Comentario 3: Añadir lemas usados al Soporte.

El siguiente teorema que se va a probar es una caracterización de las funciones inyectivas. Primero se definirá el significado de inyectividad de una función y la propiedad de ser cancelativa por la izquierda.
Una función $f : B \longrightarrow C$ es inyectiva si

$$\forall x, y \in \ B : f(x) = f(y) \Longrightarrow x = y.$$

Una función $f : B \longrightarrow C$ es cancelativa por la izquierda si

$$\forall A : (\forall g, h : X \longrightarrow Y) : f \circ g = f \circ h \Longrightarrow g = h.$$

Luego el teorema es el siguiente:

Luego el teorema es el siguiente:

Teorema 2.1.1 *f es una función inyectiva, si y solo si, para todas funciones g y h tales que $f \circ g = f \circ h$ se tiene que $g = h$.*

Vamos a hacer dos lemas de nuestro teorema, ya que se descompone la doble implicación en dos implicaciones y se va a demostrar cada una de ellas por separado.

Lema 2.1.2 (Condición necesaria) *Si f es una función inyectiva entonces para todas funciones g y h tales que $f \circ g = f \circ h$ se tiene que $g = h$.*

Demostración: Por hipótesis se tiene que $f \circ g = f \circ h$, hay que probar que $g = h$. Usando que f es inyectiva tenemos que:

$$(f \circ g)(x) = (f \circ h)(x) \Longrightarrow f(g(x)) = f(h(x)) = g(x) = h(x)$$

□

Lema 2.1.3 (Condición suficiente) *Si para toda g y h tales que $f \circ g = f \circ h$ se tiene que $g = h$ entonces f es inyectiva.*

Demostración: Si el dominio de la función f fuese vacío, f es inyectiva. Supongamos que el dominio de la función f es distinto del vacío y que f verifica la propiedad de ser cancelativa por la izquierda. Hay que demostrar que $\forall a, b$ tales que $f(a) = f(b)$, esto implica que $a = b$.
Sean a, b tales que $f(a) = f(b)$.
Definiendo $g(x) = a\ \forall x$ y $h(x) = b\ \forall x$ entonces

$$(f \circ g) = (f \circ h) \Longrightarrow f(g(x)) = f(h(x)) \Longrightarrow f(a) = f(b)$$

Por hipótesis, entonces $a = b$, como se quería demostrar.

□

2.2 Especificación en Isabelle/Hol

Su especificación es la siguiente, pero al igual que se ha hecho en la demostración a mano se va a demostrar a través de dos lemas:

theorem *caracterizacion-funcion-inyecctiva*:
 $inj\, f \longleftrightarrow (\forall\, g\, h.\ (f \circ g = f \circ h) \longrightarrow (g = h))$
 oops

Sus lemas asociados a cada implicación son los siguientes:

lemma

$\forall g\, h.\ (f \circ g = f \circ h \longrightarrow g = h) \Longrightarrow inj\, f$
oops

lemma
$inj\, f \Longrightarrow (\forall g\, h.(f \circ g = f \circ h) \longrightarrow (g = h))$
oops

En la especificación anterior, $inj\, f$ es una abreviatura de $inj\, f$ definida en la teoría Fun.thy. Además, contiene la definición de *inj-on*

$$inj\text{-}on\, f\, A = (\forall x{\in}A.\ \forall y{\in}A.\ f\, x = f\, y \longrightarrow x = y) \qquad (inj\text{-}on\text{-}def)$$

Por su parte, *UNIV* es el conjunto universal definido en la teoría Set.thy como una abreviatura de *top* que, a su vez está definido en la teoría Orderings.thy mediante la siguiente propiedad

$$\frac{ordering\text{-}top\ less\text{-}eq\ less\ top}{less\text{-}eq\ a\ top} \qquad (ordering\text{-}top.extremum)$$

En el caso de la teoría de conjuntos, la relación de orden es la inclusión de conjuntos.

Presentaremos distintas demostraciones de los lemas.

2.3 Demostración aplicativa lemas

Las demostraciones aplicativas de los lemas son :

lemma *condicion-necesaria-aplicativa*:
$inj\, f \Longrightarrow (\forall g\, h.(f \circ g = f \circ h) \longrightarrow (g = h))$
apply (*simp add*: *inj-on-def fun-eq-iff*)
done

lemma *condicion-suficiente-aplicativa*:
$\forall g\, h.\ (f \circ g = f \circ h \longrightarrow g = h) \Longrightarrow inj\, f$
apply (*rule injI*)
by (*metis fun-upd-apply fun-upd-comp*)

En las demostraciones anteriores se han usado los siguientes lemas:

$$(f = g) = (\forall x.\ f\, x = g\, x) \qquad (fun\text{-}eq\text{-}iff)$$

$$(f\,(x := y))\ z = (\text{if } z = x \text{ then } y \text{ else } f\, z) \qquad (fun\text{-}upd\text{-}apply)$$

$(f = g) = (\forall x.\, f\, x = g\, x)$ (*fun-upd-comp*)

2.4 Demostración estructurada lemas

Las demostraciones declarativas son las siguientes:

lemma *condicion-necesaria-detallada*:
  assumes $inj\, f$
  shows $\forall g\, h.(f \circ g = f \circ h) \longrightarrow (g = h)$
proof
  fix g:: $'c \Rightarrow 'a$
  show $\forall h.(f \circ g = f \circ h) \longrightarrow (g = h)$
  proof (*rule allI*)
    fix h
    show $f \circ g = f \circ h \longrightarrow (g = h)$
    proof (*rule impI*)
      assume $f \circ g = f \circ h$
      show $g = h$
      proof
        fix x
        have $(f \circ g)(x) = (f \circ h)(x)$ **using** $\langle f \circ g = f \circ h \rangle$ **by** *simp*
        then have $f(g(x)) = f(h(x))$ **by** *simp*
        thus $g(x) = h(x)$ **using** $\langle inj\, f \rangle$ **by** (*simp add*:*inj-on-def*)
      qed
    qed
  qed
qed

lemma *condicion-suficiente-detallada*:
  fixes f :: $'b \Rightarrow 'c$
  assumes $\forall (g :: 'a \Rightarrow 'b)\ (h :: 'a \Rightarrow 'b).$
      $(f \circ g = f \circ h \longrightarrow g = h)$
shows $inj\, f$
proof (*rule injI*)
  fix $a\ b$
  assume *3*: $f\, a = f\, b$
  let $?g = \lambda x :: 'a.\ a$
  let $?h = \lambda x :: 'a.\ b$
  have $\forall (h :: 'a \Rightarrow 'b).\ (f \circ ?g = f \circ h \longrightarrow ?g = h)$

using *assms* **by** $(rule\ allE)$
hence 1: $(f \circ ?g = f \circ ?h \longrightarrow ?g = ?h)$ **by** $(rule\ allE)$
have 2: $f \circ ?g = f \circ ?h$
proof
fix x
have $(f \circ (\lambda x :: 'a.\ a))\ x = f(a)$ **by** *simp*
also have ... $= f(b)$ **using** 3 **by** *simp*
also have ... $=\ (f \circ (\lambda x :: 'a.\ b))\ x$ **by** *simp*
finally show $(f \circ (\lambda x :: 'a.\ a))\ x =\ (f \circ (\lambda x :: 'a.\ b))\ x$
by *simp*
qed
have $?g = ?h$ **using** $1\ 2$ **by** $(rule\ mp)$
then show $a = b$ **by** $(rule\ fun\text{-}cong)$
qed

En la anterior demostración se ha introducito la regla:

$$\frac{(f :: 'a \Rightarrow 'b) = (g :: 'a \Rightarrow 'b)}{f\ (x :: 'a) = g\ x} \qquad (fun\text{-}cong)$$

Otras demostraciones declarativas usando auto y blast son:

lemma *condicion-necesaria-detallada1*:
assumes *inj f*
shows $(f \circ g = f \circ h) \longrightarrow (g = h)$
proof
assume $f \circ g = f \circ h$
then show $g = h$ **using** $\langle inj\ f\rangle$ **by** $(simp\ add: inj\text{-}on\text{-}def\ fun\text{-}eq\text{-}iff)$
qed

lemma *condicion-suficiente-detallada1*:
fixes $f :: 'b \Rightarrow 'c$
assumes $\forall (g :: 'a \Rightarrow 'b)\ (h :: 'a \Rightarrow 'b).$
$(f \circ g = f \circ h \longrightarrow g = h)$
shows *inj f*
proof $(rule\ injI)$
fix $a\ b$
assume 1: $f\ a = f\ b$
let $?g = \lambda x :: 'a.\ a$
let $?h = \lambda x :: 'a.\ b$
have 2: $(f \circ ?g = f \circ ?h \longrightarrow ?g = ?h)$ **using** *assms* **by** *blast*
have 3: $f \circ ?g = f \circ ?h$

proof
fix x
have $(f \circ (\lambda x :: 'a.\ a))\ x = f(a)$ **by** *simp*
also have ... $= f(b)$ **using** *1* **by** *simp*
also have ... $= (f \circ (\lambda x :: 'a.\ b))\ x$ **by** *simp*
finally show $(f \circ (\lambda x :: 'a.\ a))\ x = (f \circ (\lambda x :: 'a.\ b))\ x$
by *simp*
qed
show $a = b$ **using** *2 3* **by** *meson*
qed

2.5 Demostración teorema en Isabelle/Hol

En consecuencia, la demostración de nuestro teorema:

theorem *caracterizacion-inyectividad*:
$inj\ f \longleftrightarrow (\forall g\ h.\ (f \circ g = f \circ h) \longrightarrow (g = h))$
using *condicion-necesaria-detallada condicion-suficiente-detallada*
by *auto*

Capítulo 3

Cancelación de las funciones sobreyectivas

Comentario 4: Estructurar en secciones.

Comentario 5: Hacer demostraciones detalladas.

Comentario 6: Añadir lemas usados al Soporte.

3.1 Demostración en Lenguaje natural

El siguiente teorema prueba una caracterización de las funciones sobreyectivas. Primero se definirá el significado de la sobreyectividad de una función y de la propiedad de ser cancelativa por la derecha.
Una función $f : A \longrightarrow B$ es sobreyectiva si

$$\forall y \in B : \exists x \in A : f(x) = y$$

Una función $f : A \longrightarrow B$ tiene la propiedad de ser canletiva por la izquierda si:

$$\forall C : (\forall g, h : B \longrightarrow C) : g \circ f = h \circ f \Longrightarrow g = h$$

Luego el teorema es el siguiente:

Teorema 3.1.1 *f es sobreyectiva si y solo si para todas funciones g y h tal que $g \circ f = h \circ f$ se tiene que $g = h$.*

El teorema se puede dividir en dos lemas, ya que se demuestra por una doble implicación.

Lema 3.1.2 (Condición necesaria) *Si f es sobreyectiva entonces para todas funciones g y h tal que $g \circ f = h \circ f$ se tiene que $g = h$.*

Demostración: Supongamos que tenemos que $g \circ f = h \circ f$, queremos probar que $g = h$. Usando la definición de sobreyectividad $(\forall y \in Y, \exists x \| y = f(x))$ y nuestra hipótesis, tenemos que:

$$g(y) = g(f(x)) = (g \circ f)(x) = (h \circ f)(x) = h(f(x)) = h(y).$$

$\square$

Lema 3.1.3 (Condición necesaria) *Si para todas funciones g y h tal que $g \circ f = h \circ f$ se tiene que g = h entonces f es sobreyectiva.*

Demostración: Para la demostración del lema, primero se debe señalar los dominios y codominios de las funciones que se van a usar. $f : C \longrightarrow A$, $g, h : A \longrightarrow B$. También se debe notar que el conjunto B tiene que tener almenos dos elementos diferentes,luego supongamos que $B = \{a, b\}$.
La prueba se va a realizar por reducción al absurdo. Luego supongamos que nuestra función f no es sobreyectiva, es decir, $\exists y_1 \in A$ *tal que* $\nexists x \in C \ : f(x) = y$.
Definamos ahora las funciones g, h :

$$g(y) = a \; \forall y \in A$$

$$h(y) = \begin{cases} a & si \quad y \neq y_1 \\ b & si \quad y = y_1 \end{cases}$$

Entonces $g(y) \neq h(y)$. Sin embargo, por hipótesis se tiene que si $g \circ f = h \circ f$, lo cual es cierto, entonces $h = g$. Por lo que hemos llegado a una contradicción, por lo tanto, f es sobreyectiva.

$\square$

3.2 Especificación en Isabelle/Hol

Su especificación es la siguiente, que se dividira en dos al igual que en la demostración a mano:

theorem *caracterizacion-funciones-sobreyectivas*:
$surj\, f \longleftrightarrow (\forall\, g\, h.(g \circ f = h \circ f) \longrightarrow (g = h))$
oops

lemma *condicion-suficiente*:
$surj\, f \Longrightarrow\ (\forall\, g\, h.\ (g \circ f = h \circ f) \longrightarrow (g = h))$
oops

lemma *condicion-necesaria*:
$\forall\, g\, h.\ (g \circ f = h \circ f \longrightarrow g = h) \longrightarrow surj\, f$
oops

En la especificación anterior, $surj\, f$ es una abreviatura de $range\, f = UNIV$, donde $range\, f$ es el rango o imagen de la función f y $UNIV$ es el conjunto universal definido en la teoría Set.thy como una abreviatura de *top* que, a su vez está definido en la teoría Orderings.thy mediante la siguiente propiedad

$$\frac{ordering\text{-}top\ less\text{-}eq\ less\ top}{less\text{-}eq\ a\ top} \qquad (ordering\text{-}top.extremum)$$

Además queda añadir que la teoría donde se encuentra definido $surj\, f$ es en Fun.thy. Esta teoría contiene la definicion *surj-def*.

$$surj\, f = (\forall\, y.\ \exists\, x.\ y = f\, x) \qquad (surj\text{--}def)$$

3.3 Demostración estructurada

Presentaremos distintas demostraciones de los lemas. Las primeras son las detalladas:

lemma *condicion-suficiente-detallada*:
 assumes $surj\, f$
 shows $\forall\, g\, h.\ (\ g \circ f = h \circ f\ ) \longrightarrow (g = h)$
proof $(rule\ allI)$
 fix $g :: 'a \Rightarrow 'c$
 show $\forall\, h.\ (g \circ f = h \circ f) \longrightarrow (g = h)$
 proof $(rule\ allI)$
  fix h
  show $(g \circ f = h \circ f) \longrightarrow (g = h)$
  proof $(rule\ impI)$

```
    assume 1: g ∘ f = h ∘ f
    show g = h
    proof
     fix x
     have ∃ y . x = f(y) using assms by (simp add:surj-def)
     then obtain y where 2:x = f(y) by (rule exE)
     then have g(x) = g(f(y)) by simp
     also have ... = (g ∘ f) (y)  by simp
     also have ... = (h ∘ f) (y) using 1 by simp
     also have ... = h(f(y)) by simp
     also have ... = h(x) using 2  by (simp add: ‹x = f y›)
     finally show  g(x) = h(x)  by simp
    qed
   qed
  qed
qed
```

En la siguiente demostración nos hará falta la introducción de los pequeños lemas que demostraremos a continuación:

```
lemma auxiliar-1:
  assumes ¬(∀ x. P(x))
  shows  ∃ x. ¬P(x)
using assms
  by auto

lemma auxiliar-2:
  assumes ¬(∃ x. P(x))
  shows  ∀ x. ¬P(x)
using assms
by auto

lemma condicion-necesaria-detallada:
  assumes ∀ (g :: 'b ⇒ 'c) h .(g ∘ (f :: 'a ⇒ 'b) = h ∘ f) ⟶ (g = h)
       ∃ (x0::'c) (x1::'c). x0 ≠ x1
      shows ∀ (y::'b). (∃ (x:: 'a). f x = y)
proof (rule ccontr)
  assume ¬ (∀ y :: 'b. ∃ x :: 'a. f x = y)
  hence ∃ y :: 'b . ¬ (∃ x :: 'a. f x = y) by (rule auxiliar-1)
  then obtain y0 where  ¬ (∃ x :: 'a. f x = y0) by (rule exE)
  hence ∀ x :: 'a. (¬ (f x = y0)) by (rule auxiliar-2)
  obtain a0 where  ∃ (x1::'c). a0 ≠ x1 using assms(2) by (rule exE)
```

then obtain $a1$ **where** $a0 \neq a1$ **by** $(rule\ exE)$
let $?g = (\lambda x.\ a0)\ ::\ 'b \Rightarrow\ 'c$
let $?h = ?g(y0{:=}a1)$
have $\forall\, h\ .(?g \circ (f\ ::\ 'a \Rightarrow\ 'b) = h \circ f) \longrightarrow (?g = h)$
using $assms(1)$ **by** $(rule\ allE)$
hence $1{:}(?g \circ (f\ ::\ 'a \Rightarrow\ 'b) = ?h \circ f) \longrightarrow (?g = ?h)$ **by** $(rule\ allE)$
have $2{:}\ (?g \circ (f\ ::\ 'a \Rightarrow\ 'b) = ?h \circ f)$
using $[[simp\text{-}trace]]$
using $\langle \nexists\, x\ ::\ 'a.\ (f\ ::\ 'a \Rightarrow\ 'b)\ x = (y0\ ::\ 'b)\rangle$ **by** *auto*
have $(?g = ?h)$ **using** $1\ 2$ **by** $(rule\ mp)$
hence $a0 = a1$ **by** $(metis\ fun\text{-}upd\text{-}idem\text{-}iff)$
with $\langle a0 \neq a1\rangle$ **show** *False* **by** $(rule\ notE)$
qed

lemma *condicion-necesaria-detallada-2*:
assumes $\forall\ (y{::}'b).\ (\exists\ (x{::}\ 'a).\ f\, x = y)$
shows $surj\ f$
by $(metis\ assms\ surj\text{-}def)$

En la demostración hemos introducido:

$$\frac{\exists\, x.\ P\, x \qquad \bigwedge x.\ \dfrac{P\, x}{Q}}{Q} \tag{rule exE}$$

$$[\![P \Longrightarrow Q;\ Q \Longrightarrow P]\!] \Longrightarrow P = Q \tag{iffI}$$

3.4 Demostración aplicativa

Las demostraciones aplicativas son:

lemma *demostracion-suficiente-aplicativa*:
$surj\ f \Longrightarrow ((g \circ f) = (h \circ f)\ ) \longrightarrow (g = h)$
apply $(simp\ add{:}\ surj\text{-}def\ fun\text{-}eq\text{-}iff)$
apply *metis*
done

lemma *demostracion-necesaria-aplicativa*:
$[\![\forall\, (g\ ::\ 'b \Rightarrow\ 'c)\ h\ .(g \circ (f\ ::\ 'a \Rightarrow\ 'b) = h \circ f) \longrightarrow (g = h);$

$\exists\ (x0{::}'c)\ (x1{::}'c).\ x0 \neq x1 \rrbracket \Longrightarrow surj\ f$
apply (*rule surjI*)
apply (*drule condicion-necesaria-detallada*)
apply *simp*
apply (*erule allE*)
apply (*erule exE*)+
apply (*erule Hilbert-Choice.someI*)
done

En estas hemos introducido:

$$(f = g) = (\forall\, x.\, f\, x = g\, x) \qquad (fun\text{-}eq\text{-}iff)$$

$$\frac{P\ x}{P\ (Eps\ P)} \qquad (Hilbert\text{-}Choice.someI)$$

3.5 Demostración teorema

En consecuencia, la demostración del teorema es

theorem *caracterizacion-funciones-sobreyectivas*:
$surj\ f \longleftrightarrow\ (\forall\, g\ h.(g \circ f = h \circ f) \longrightarrow (g = h))$
oops

Capítulo 4

Propiedad de los conjuntos finitos de números naturales

Comentario 7: Estructurar en secciones.

Comentario 8: Hacer demostraciones detalladas.

Comentario 9: Añadir lemas usados al Soporte.

4.1 Demostración en lenguaje natural

El siguiente teorema es una propiedad que verifican todos los conjuntos finitos de números naturales estudiado en el tema 10 de la asignatura de LMF de tercer curso del grado en Matemáticas. Su enunciado es el siguiente

Teorema 4.1.1 *Sea S un conjunto finito de números naturales. Entonces todos los elementos de S son menores o iguales que la suma de los elementos de S, es decir,*

$$\forall m \in S \Longrightarrow m \leq \sum S$$

donde $\sum S$ denota la suma de todos los elementos de S.

Demostración: La demostración del teorema la haremos por inducción sobre conjuntos finitos.

(Base de la inducción) El caso $S = \emptyset$ es trivial.

(Paso de la inducción) Supongamos que se verifica el teorema para un conjunto finito de números naturales, que se denotará por S.

Sea $a \in \mathbb{N}$ tal que $a \notin S$, Ya que si $a \in S$ se tendría probado el teorema. Luego hay que probar que:

$$\forall n \in S \cup \{a\} \Longrightarrow n \leq \sum(S \cup \{a\})$$

Distingamos dos casos ahora:

Caso 1: $n = a$.

Si $n = a$, se tiene que:

$$n = a \leq a + \sum S = \sum(S \cup \{a\}).$$

Caso 2: $n \neq a$.

Si $n \neq a$, tenemos que $n \in S$, luego usando la hipótesis de inducción:

$$n \leq \sum S \leq \sum S + a = \sum(S \cup \{a\}).$$

□

En la demostración del teorema hemos usado un resultado, que vamos a probar en Isabelle después de la especificación del teorema, el resultado es $\sum S + a = \sum(S \cup \{a\})$.

4.2 Especificación en Isabelle/HOL

Para la especificación del teorema en Isabelle, primero debemos notar que *finite S* indica que un conjunto S es finito y definir la función *sumaConj* tal que *sumaConj n* es la suma de todos los elementos de S.

definition *sumaConj* :: *nat set* ⇒ *nat* **where**
sumaConj S ≡ $\sum S$

El enunciado del teorema es el siguiente :

lemma *finite S* ⟹ ∀ *x* ∈ *S*. *x* ≤ *sumaConj S*

oops

Vamos a demostrar primero el lema enunciado anteriormente

lemma *aux-propiedad-conjuntos-finitos*:
 $x \notin S \wedge finite\ S \longrightarrow sumaConj\ S + x = sumaConj(insert\ x\ S)$
 by (*simp add*: *sumaConj-def*)

La demostración del lema anterior se ha incluido *sumConj-def*, que hace referencia a la definición sumaConj que hemos hecho anteriormente.

En la demostración se usará la táctica *induct* que hace uso del esquema de inducción sobre los conjuntos finitos:

$$[\![finite\ x;\ P\ \emptyset;\ \bigwedge A\ a.\ finite\ A \wedge P\ A \Longrightarrow P\ (\{a\} \cup A)]\!] \Longrightarrow P\ x \qquad (finite.induct)$$

Vamos a presentar diferentes formas de demostración:

4.3 Demostración aplicativa

La demostración aplicativa del teorema es:

lemma $finite\ S \Longrightarrow \forall x \in S.\ x \leq sumaConj\ S$
 apply (*induct rule*: *finite-induct*)
  apply *simp*
 apply (*simp add*: *add-increasing sumaConj-def*)
 done

En la demostración anterior se ha introducido:

$$\frac{(0 :: 'a) \leq a \wedge b \leq c}{b \leq a + c} \qquad (add\text{-}increasing)$$

4.4 Demostración automática

La demostración automática es:

lemma $finite\ S \Longrightarrow \forall x \in S.\ x \leq sumaConj\ S$
 by (*induct rule*: *finite-induct*)
  (*auto simp add*: *sumaConj-def*)

4.5 Demostración detallada

La demostración declarativa es:

lemma *sumaConj-acota*:
 finite S $\Longrightarrow \forall x{\in}S.\ x \leq$ *sumaConj S*
proof (*induct rule*: *finite-induct*)
 show $\forall x \in \{\}.\ x \leq$ *sumaConj* $\{\}$ **by** *simp*
next
 fix *x* **and** *F*
 assume *fF*: *finite F*
  and *xF*: $x \notin F$
  and *HI*: $\forall\ x{\in}F.\ x \leq$ *sumaConj F*
 show $\forall y \in$ *insert x F*. $y \leq$ *sumaConj* (*insert x F*)
 proof
  fix *y*
  assume $y \in$ *insert x F*
  show $y \leq$ *sumaConj* (*insert x F*)
  proof (*cases* $y = x$)
   assume $y = x$
   then have $y \leq x +$ (*sumaConj F*) **by** *simp*
   also have $\ldots =$ *sumaConj* (*insert x F*)
    by (*simp add*: *fF sumaConj-def xF*)
   finally show *?thesis* **.**
  next
   assume $y \neq x$
   then have $y \in F$ **using** ⟨$y \in$ *insert x F*⟩ **by** *simp*
   then have $y \leq$ *sumaConj F* **using** *HI* **by** *simp*
   also have $\ldots \leq x +$ (*sumaConj F*) **by** *simp*
   also have $\ldots =$ *sumaConj* (*insert x F*) **using** *fF xF*
    by (*simp add*: *sumaConj-def*)
   finally show *?thesis* **.**
  qed
 qed
qed

En esta última demostración hemos usado el método de prueba por casos,el método blast y también el simp(”simplificador”) añadiéndole *sumaConj-def*.

Capítulo 5

Teorema de Cantor

Comentario 10: Estructurar en secciones.

Comentario 11: Hacer demostraciones detalladas.

Comentario 12: Añadir lemas usados al Soporte.

El siguiente, denominado teorema de Cantor por el matemático Georg Cantor, es un resultado importante de la teoría de conjuntos.

El matemático Georg Ferdinand Ludwig Philipp Cantor fue un matemático y lógico nacido en Rusia en el siglo XIX. Fue inventor junto con Dedekind y Frege de la teoría de conjuntos, que es la base de las matemáticas modernas.

Para la comprensión del teorema vamos a definir una serie de conceptos:

- Conjunto de potencia A ($\mathcal{P}(A)$): conjunto formado por todos los subconjuntos de A.
- Cardinal del conjunto A (Denotado #A): número de elementos del propio conjunto.

El enunciado original del teorema es el siguiente :

Teorema 5.0.1 *El cardinal del conjunto potencia de cualquier conjunto A es estrictamente mayor que el cardinal de A, o lo que es lo mismo, $\#\mathcal{P}(A) > \#A$.*

Pero el enunciado del teorema lo podemos reformular como:

Teorema 5.0.2 *Dado un conjunto A,* $\nexists f : A \longrightarrow \mathcal{P}(A)$ *que sea sobreyectiva.*

El teorema lo hemos podido reescribir de la anterior forma, ya que si suponemos que $\exists f$ tal que $f : A \longrightarrow \mathcal{P}(A)$ es sobreyectiva, entonces tenemos que $f(A) = \mathcal{P}(A)$ y por lo tanto, $\#f(A) \geq \#\mathcal{P}(A)$, de lo que se deduce esta reformulación. Reciprocamente, es trivial ver que esta reformulación implica la primera. con el teorema.
El teorema de Cantor es trivial para conjuntos finitos, ya que el conjunto potencia, de conjuntos finitos de n elementos tiene 2^n elementos.

Por ello, vamos a realizar la prueba para conjuntos infinitos.

Demostración:

Vamos a realizar la prueba por reducción al absurdo.
Supongamos que $\exists f : A \longrightarrow \mathcal{P}(A)$ sobreyectiva, es decir, $\forall C \in \rho(A), \exists x \in A$ tal que $C = f(x)$. En particular, tomemos el conjunto

$$B = \{x \in A : x \notin f(x)\}$$

y supongamos que $\exists a \in A : B = f(a)$, ya que B es un subconjunto de A, luego podemos distinguir dos casos :
1. Si $a \in B$, entonces por definición del conjunto B tenemos que $a \notin B$, luego llegamos a una contradicción.
2. Si $a \notin B$, entonces por definición de B tenemos que $a \in B$, luego hemos llegado a otra contradicción.

En las dos hipótesis hemos llegado a una contradicción, por lo que no existe a y f no es sobreyectiva.

□

Para la especificación del teorema en Isabelle, primero debemos notar que

$$f :: {}'a \Rightarrow {}'a\, set$$

significa que es una función de tipos, donde $'a$ significa un tipo y para poder denotar el conjunto potencia tenemos que poner $'a\ set$ que significa que es de un tipo formado por conjuntos del tipo $'a$.

El enunciado del teorema es el siguiente :

theorem *Cantor*: $\nexists f :: 'a \Rightarrow 'a\ set.\ \forall A.\ \exists x.\ A = f\, x$

oops

La demostración la haremos por la regla la introducción a la negación, la cual es una simplificación de la regla de reducción al absurdo, cuyo esquema mostramos a continuación:

$(P \Longrightarrow False) \Longrightarrow \neg P$ (*notI*)

Esta es la demostración detallada del teorema:

theorem *CantorDetallada*: $\nexists f :: 'a \Rightarrow 'a\ set.\ \forall B.\ \exists x.\ B = f\,x$
proof (*rule notI*)
 assume $\exists f :: 'a \Rightarrow 'a\ set.\ \forall A.\ \exists x.\ A = f\,x$
 then obtain $f :: 'a \Rightarrow 'a\ set$ **where** $*$: $\forall A.\ \exists x.\ A = f\,x$ **by** (*rule exE*)
 let $?B = \{x.\ x \notin f\,x\}$
 from $*$ **obtain** $\exists x.\ ?B = f\,x$ **by** (*rule allE*)
 then obtain *a* **where** 1:$?B = f\,a$ **by** (*rule exE*)
 show *False*
 proof (*cases*)
  assume $a \in ?B$
  then show *False* **using** *1* **by** *blast*
 next
  assume $a \notin ?B$
  thus *False* **using** *1* **by** *blast*
 qed
qed

Esta es la demostración aplicativa del teorema:

theorem *CantorAplicativa* :
$\nexists f :: 'a \Rightarrow 'a\ set.\ \forall A.\ \exists x.\ A = f\,x$
 apply (*rule notI*)
 apply (*erule exE*)
 apply (*erule-tac* $x = \{x.\ x \notin f\,x\}$ **in** *allE*)
 apply (*erule exE*)
 apply *blast*
 done

Esta es la demostración automática del teorema:

theorem *CantorAutomatic*: $\nexists f :: 'a \Rightarrow 'a\ set.\ \forall B.\ \exists x.\ B = f\,x$
 by *best*

En la demostración de isabelle hemos utilizado el método de prueba rule con las siguientes reglas, tanto en la aplicativa como en la detallada:

$$\frac{\dfrac{P}{False}}{\neg P} \qquad (notI)$$

$$\frac{\exists\, x.\; P\,x \qquad \bigwedge x.\; \dfrac{P\,x}{Q}}{Q} \tag{exE}$$

$$\frac{\forall\, x.\; P\,x \qquad \dfrac{P\,x}{R}}{R} \tag{allE}$$

También hacemos uso de blast, que es un conjunto de reglas lógicas y la demostración automática la hacemos por medio de ”best”.

Apéndice A

Métodos de pruebas y reglas

Métodos de pruebas de demostraciones:

$$[\![P\ 0;\ \bigwedge nat.\ P\ nat \Longrightarrow P\ (Suc\ nat)]\!] \Longrightarrow P\ nat \qquad (nat.induct)$$

$$[\![P \Longrightarrow Q;\ Q \Longrightarrow P]\!] \Longrightarrow P = Q \qquad (iffI)$$

$$[\![finite\ x;\ P\ \emptyset;\ \bigwedge A\ a.\ finite\ A \wedge P\ A \Longrightarrow P\ (\{a\} \cup A)]\!] \Longrightarrow P\ x \qquad (finite.induct)$$

$$(P \Longrightarrow False) \Longrightarrow \neg\ P \qquad (notI)$$

Reglas usadas:

$$inj\text{-}on\ f\ A = (\forall\ x{\in}A.\ \forall\ y{\in}A.\ f\ x = f\ y \longrightarrow x = y) \qquad (inj\text{-}on\text{-}def)$$

$$\frac{ordering\text{-}top\ less\text{-}eq\ less\ top}{less\text{-}eq\ a\ top} \qquad (ordering\text{-}top.extremum)$$

$$(f = g) = (\forall\ x.\ f\ x = g\ x) \qquad (fun\text{-}eq\text{-}iff)$$

$$(f \circ g)\ x = f\ (g\ x) \qquad (o\text{-}apply)$$

$$\frac{\dfrac{P}{Q} \qquad \dfrac{Q}{P}}{P = Q} \qquad (iffI)$$

$$\frac{\mathit{ListMem}\ x\ xs}{\mathit{ListMem}\ x\ (y \cdot xs)} \tag{insert}$$

$$\frac{\exists\, x.\ P\ x \qquad \bigwedge x.\ \dfrac{P\ x}{Q}}{Q} \tag{exE}$$

$$\frac{\forall\, x.\ P\ x \qquad \dfrac{P\ x}{R}}{R} \tag{allE}$$

$$\frac{\dfrac{P}{\mathit{False}}}{\neg\, P} \tag{notI}$$

$$((P \longrightarrow Q) \wedge (\neg\, P \longrightarrow Q)) = Q \tag{cases}$$

Apéndice B

Lemas de HOL usados

En este apéndice se recogen la lista de los lemas usados en el trabajo indicando la página del libro de HOL donde se encuentra.

Comentario 13: Añadir el libro de HOL a la bibliografía.

Comentario 14: Completar la lista de lemas usados.

B.1 Números naturales (16)

B.1.1 Operaciones aritméticas (16.3)

- (p. 348) $0 * n = 0$ (*mult-0*)
- (p. 348) $Suc\ m * n = n + m * n$ (*mult-Suc*)
- (p. 348) $m * Suc\ n = m + m * n$ (*mult-Suc-right*)

Bibliografía

[1] José A. Alonso. Temas de "Lógica matemática y fundamentos (2018–19)". Technical report, Univ. de Sevilla, 2019. En https://www.cs.us.es/~jalonso/cursos/lmf-18/temas.php.

[2] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A proof assistant for Higher–Order Logic*. Lecture Notes in Computer Science, Vol. 2283, Springer–Verlag, 2019. En https://www.cl.cam.ac.uk/research/hvg/Isabelle/dist/Isabelle2019/doc/tutorial.pdf.

Lista de tareas pendientes