Formalización en Lean de soluciones del IMO

Sara Díaz Real

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, 3 de febrero de 2021

Índice general

Resumen

En este capítulo se describe el TFM (como se realizará en la presentación del TFM). Es el último que se escribe.

Introducción

En este capítulo se realiza una introducción a los sistemas usados para que la memoria sea autocontenida.

Capítulo 1

Elementos de matemáticas en Lean

En este capítulo se presentan algunas demostraciones de las que se estudian en diversas asignaturas del Grado de Matemáticas formalizadas en Lean.

1.1. Cálculo infinitesimal

En esta primera sección veremos diferentes resultados de análisis correspondientes a la asignatura de formación básica Cálculo Infinitesimal del primer año del Grado.

Estudiaremos un total de seis resultados. Los tres primeros son relativos a la convergencia de sucesiones, mientras que los tres últimos son sobre la paridad de las funciones.

1.1.1. Unicidad del límite

Definición 1.1.1. Una sucesión $\{u_n\}$ se dice que converge a un número real a si, dado cualquier número real $\epsilon > 0$, existe un número natural N tal que si n es cualquier número natural mayor o igual que N se cumple que $|u_n - a| \leq \epsilon$. Simbólicamente:

$$\forall \epsilon > 0, \exists N \in \mathbb{N}, \forall n \geq N,\ |u_n - a| \leq \epsilon. \tag{1.1}$$

La formalización de la definición anterior en Lean es

```
def limite : (ℕ → ℝ) → ℝ → Prop :=
λ u c, ∀ ε > 0, ∃ N, ∀ n ≥ N, |u n - c| ≤ ε
```

donde la notación para el valor absoluto se ha definido por

```
notation `|`x`|` := abs x
```

Teorema 1.1.2. Cada sucesión tiene como máximo un límite.

Demostración: Comenzaremos viendo la demostración en lenguaje natural. Para ello, procederemos por Reducción al Absurdo: supondremos que la sucesión $\{u_n\}$ posee dos límites distintos que denotaremos por l y l'. Tenemos que demostrar que estos límites son iguales, es decir, que $l = l'$.

En primer lugar, por la definición de límite anterior, 1.1.1, se tiene que para el límite l podemos escribir que:

$$\forall \epsilon > 0, \exists N_1 \in \mathbb{N},\ \forall n \geq N_1 |u_n - l| \leq \frac{\epsilon}{2}. \tag{1.2}$$

Análogamente, para l' se tiene que:

$$\forall \epsilon > 0, \exists N_2 \in \mathbb{N},\ \forall n \geq N_2 |u_n - l'| \leq \frac{\epsilon}{2}. \tag{1.3}$$

A continuación, definamos $N_0 = \max(N_1, N_2)$. De manera que considerando un número arbitrario n que sea mayor o igual que N_0 y teniendo en cuenta lo visto anteriormente, podemos escribir:

$$|l - l'| = |l - u_{N_0} + u_{N_0} - l'| \overset{(*)}{\leq} |u_{N_0} - l| + |u_{N_0} - l'| \overset{(**)}{\leq} \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon,$$

donde en $(*)$ se ha hecho uso de la desigualdad triangular y la definición de valor absoluto y en $(**)$ se ha usado las desigualdades (1.2) y (1.3).

Por tanto, se ha demostrado que para todo ϵ mayor que cero se verifica que

$$|l - l'| \leq \epsilon. \tag{1.4}$$

Finalmente, se concluiría la demostración sin más que aplicar sobre (1.4) el siguiente resultado:

$$x, y \in \mathbb{R},\ \forall \epsilon > 0,\ |x - y| \leq \epsilon \implies x = y. \tag{1.5}$$

□

La formalización en Lean del teorema anterior es

```
import data.real.basic

variables (u : ℕ → ℝ)
variables (a b x y : ℝ)

notation `|`x`|` := abs x

def limite : (ℕ → ℝ) → ℝ → Prop :=
λ u c, ∀ ε > 0, ∃ N, ∀ n ≥ N, |u n - c| ≤ ε

lemma eq_of_abs_sub_le_all
: (∀ ε > 0, |x - y| ≤ ε) → x = y :=
begin
intro h,
apply eq_of_abs_sub_nonpos,
by_contradiction H,
push_neg at H,
specialize h ( |x-y|/2) (by linarith),
linarith,
end

example
(ha : limite u a)
(hb : limite u b)
: a = b :=
begin
apply eq_of_abs_sub_le_all,
intros eps eps_pos,
cases ha (eps/2) (by linarith) with N1 hN1,
cases hb (eps/2) (by linarith) with N2 hN2,
let N0:=max N1 N2,
calc  |a-b|
    = |(a-u N0)+(u N0-b)| : by ring
... ≤ |a-u N0| + |u N0-b| : by apply abs_add
... = |u N0-a| + |u N0-b| : by rw abs_sub
... ≤ eps                 : by linarith [hN1 N0 (le_max_left N1 N2),
                                         hN2 N0 (le_max_right N1 N2)],
end
```

1.1.2. Las sucesiones convergentes son sucesiones de Cauchy

Previo al enunciado y demostración del Teorema, veamos en qué consiste una sucesión de Cauchy.

Definición 1.1.3. Una sucesión $\{u_n\}$ se dice que es de Cauchy si, dado cualquier número real $\epsilon > 0$, existe un número natural N tal que si p y q son cualesquiera dos números naturales mayores o iguales que N se cumple que $|u_p - u_q| \leq \epsilon$. Simbólicamente:

$$\forall \epsilon > 0,\ \exists N \in \mathbb{N},\ \forall p, q \geq N,\ |u_p - u_q| \leq \epsilon. \tag{1.6}$$

En Lean esta definición se formaliza como:

```
def cauchy_sequence (u : ℕ → ℝ) :=
∀ ε > 0, ∃ N, ∀ p q, p ≥ N → q ≥ N → |u p - u q| ≤ ε
```

Teorema 1.1.4. Toda sucesión convergente es una sucesión de Cauchy.

Demostración: Sea $\{u_n\}$ una sucesión convergente, es decir, por la definición 1.1.1, se tiene que si denotamos l como el límite de la sucesión en cuestión se verifica que

$$\forall \epsilon > 0, \exists N \in \mathbb{N},\ \forall n \geq N\ |u_n - l| \leq \frac{\epsilon}{2}. \tag{1.7}$$

A continuación, consideremos $p, q \in \mathbb{N}$ tales que estos dos números verifican que son mayores o iguales que N. Entonces tenemos que se puede escribir:

$$|u_p - u_q| = |u_p - l + l - u_q| \overset{(*)}{\leq} |u_p - l| + |u_q - l| \overset{(**)}{=} \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon,$$

donde $(*)$ se ha hecho uso de la desigualdad triangular y de la propia definición del valor absoluto; mientras que en $(**)$se ha usado lo visto en (1.7) para los casos de p y q.

De esta forma, se ha demostrado que para cualquier número real $\epsilon > 0$ se verifica que $|u_p - u_q| \leq 0$, es decir, hemos demostrado que $\{u_n\}$ es una sucesión de Cauchy.

□

En Lean esto se formalice como sigue:

```
import data.real.basic

variables {u : ℕ → ℝ} {a l : ℝ}

notation `|`x`|` := abs x

def seq_limit (u : ℕ → ℝ) (l : ℝ) : Prop :=
∀ ε > 0, ∃ N, ∀ n ≥ N, |u n - l| ≤ ε

def cauchy_sequence (u : ℕ → ℝ) :=
∀ ε > 0, ∃ N, ∀ p q, p ≥ N → q ≥ N → |u p - u q| ≤ ε

example : (∃ l, seq_limit u l) → cauchy_sequence u :=
begin
intros h eps eps_pos,
cases h with l hl,
rw seq_limit at hl,
cases hl (eps/2) (by linarith) with N hN,
use N,
  intros p q hp hq,
calc  |u p - u q|
    = |(u p - l)+(l - u q)|  : by ring
... ≤ |u p - l| + |l - u q|  : by apply abs_add
... = |u p - l| + |u q - l|  : by rw abs_sub l (u q)
... ≤ eps/2 + eps/2          : by linarith [hN p hp, hN q hq]
... = eps                    : by ring,
end
```

1.1.3. Suma de sucesiones convergentes

Teorema 1.1.5. Sean $\{u_n\}$ y $\{v_n\}$ dos sucesiones convergentes, entonces se tiene que la suma de estas dos sucesiones converge a la suma de sus respectivos límites. Simbólicamente:

$$lim(u_n + v_n) = lim(u_n) + lim(v_n). \tag{1.8}$$

Demostración: Como consecuencia de que las sucesiones $\{u_n\}$ y $\{v_n\}$ sean convergentes se tiene que:

$$\forall \epsilon > 0, \exists N_1 \in \mathbb{N},\ \forall n \geq N_1\ |u_n - l| \leq \frac{\epsilon}{2} \qquad \text{y} \tag{1.9}$$

$$\forall \epsilon > 0, \exists N_2 \in \mathbb{N},\ \forall n \geq N_1\ |v_n - l'| \leq \frac{\epsilon}{2}, \tag{1.10}$$

donde hemos denotado l y l' a los límites de $\{u_n\}$ y $\{v_n\}$, respectivamente.

A continuación, definimos $N_0 := max(N_1, N_2)$ y consideremos un número natural n que sea mayor o igual que N_0, de manera que:

$$|(u_n + v_n) - (l + l')| \overset{(*)}{\leq} |u_n - l| + |v_n - l'| \overset{(**)}{\leq} \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon,$$

donde en $(*)$ se usado la desigualdad triangular y en $(**)$ las desigualdades vistas en (1.9) y (1.10).

De manera que finalmente se ha demostrado que para cualquier número real $\epsilon > 0$, existe un número natural N_0 tal que para cualquier número n mayor o igual que N_0, se verifica que $|(u_n + v_n) - (l + l')| \leq \epsilon$, o lo que es lo mismo que la sucesión $\{u_n + v_n\}$ converge a $(l + l')$.

□

En Lean esto se formaliza como:

```
import data.real.basic
import algebra.group.pi

variables {u v: ℕ → ℝ} {l l' : ℝ}

notation `|`x`|` := abs x

def seq_limit (u : ℕ → ℝ) (l : ℝ) : Prop :=
∀ ε > 0, ∃ N, ∀ n ≥ N, |u n - l| ≤ ε

lemma ge_max_iff {α : Type*} [linear_order α] {p q r : α} :
r ≥ max p q  ↔ r ≥ p ∧ r ≥ q :=
max_le_iff

example (hu : seq_limit u l) (hv : seq_limit v l') :
seq_limit (u + v) (l + l') :=
begin
intros eps eps_pos,
cases hu (eps/2) (by linarith) with N1 hN1,
cases hv (eps/2) (by linarith) with N2 hN2,
let N0:= max N1 N2,
use N0,
intros n hn,
rw ge_max_iff at hn,
  calc |(u + v) n - (l + l')|
     = |u n + v n - (l + l')|   : rfl
 ... = |(u n - l) + (v n - l')| : by congr' 1 ; ring
 ... ≤ |u n - l| + |v n - l'|   : by apply abs_add
```

```
 ... ≤  eps                        : by linarith [hN1 n (by linarith),
                                                   hN2 n (by linarith)],
end
```

1.1.4. Paridad de la suma de funciones

Como ya se adelantó, los tres resultados que vamos a estudiar a continuación son relativos a la paridad de las funciones. Es por eso, que previo al desarrollo de los resultados veamos las dos siguientes definiciones:

Definición 1.1.6. Sea $f : \mathbb{R} \to \mathbb{R}$, se dirá que f es una función par si para cualquier número real x se verifica que $f(-x) = f(x)$.

Definición 1.1.7. Sea $f : \mathbb{R} \to \mathbb{R}$, se dirá que f es una función impar si para cualquier número real x se verifica que $f(-x) = -f(x)$.

Estas dos definiciones se formalizan en Lean como sigue:

```
def even_fun (f : ℝ → ℝ) := ∀ x, f (-x) = f x
def odd_fun (f : ℝ → ℝ) := ∀ x, f (-x) = -f x
```

A continuación, ya estamos preparados para ver la demostración del resultado deseado.

Teorema 1.1.8. Sean f y g dos funciones pares, entonces se tiene que la suma de ambas, $(f + g)$, es también una función par.

Demostración: Por la definición 1.1.6 de una función par, se tiene que se verifica lo siguiente:

$$f(-x) = f(x) \qquad \forall x \in \mathbb{R} \quad \text{y} \tag{1.11}$$

$$g(-x) = g(x) \qquad \forall x \in \mathbb{R}. \tag{1.12}$$

Como queremos demostrar que la suma de las funciones f y g es una función par, se tendría que probar que

$$(f + g)(-x) = (f + g)(x) \qquad \forall x \in \mathbb{R}. \tag{1.13}$$

Para probarlo, comencemos considerando un número real x arbitrario, entonces estudiemos:

$$(f + g)(-x) \overset{(1)}{=} f(-x) + g(-x) \overset{(2)}{=} f(x) + g(x) \overset{(3)}{=} (f + g)(x),$$

donde tanto en el paso (1) como en el (3) hemos usado la propia definición de la suma de funciones, mientras que en (2) se han usado las hipótesis de paridad (1.11) y (1.12). Por tanto, ya tendríamos el resultado deseado.

□

Esta demostración se formaliza en Lean de la siguiente manera:

```
import data.real.basic
import algebra.group.pi

def even_fun (f : ℝ → ℝ) := ∀ x, f (-x) = f x

example (f g : ℝ → ℝ) : even_fun f → even_fun g →  even_fun (f + g) :=
begin
intros hf hg,
intros x,
calc (f + g) (-x)
    = f (-x) + g (-x) : rfl
... = f x + g (-x)    : by rw hf
... = f x + g x       : by rw hg
... = (f + g) x       : rfl
end
```

1.1.5. Paridad de la composición de funciones

Teorema 1.1.9. Sea f una función par y sea g una función arbitraria, entonces se tiene que la composición de g con f, es decir, $(g \circ f)$ es también par.

Demostración: Al igual que en la demostración anterior, por la definición 1.1.6, se tiene que para la función f se verifica que

$$f(-x) = f(x) \qquad \forall x \in \mathbb{R}. \tag{1.14}$$

Consideremos ahora un número real x arbitrario y estudiemos la composición de una función arbitraria g con f:

$$(g \circ f)(-x) \overset{(1)}{=} g(f(-x)) \overset{(2)}{=} g(f(x)) \overset{(3)}{=} (g \circ f)(x),$$

donde en (1) y (3) se ha usado la definición de la composición; mientras que en (2) se ha hecho uso de la hipótesis de paridad de f, (1.14).

□

En Lean esto se formalizaría como sigue:

```
import data.real.basic
import algebra.group.pi

def even_fun (f : ℝ → ℝ) := ∀ x, f (-x) = f x

example (f g : ℝ → ℝ) : even_fun f → even_fun (g ∘ f) :=
begin
intros hf x,
calc (g ∘ f) (-x)
    = g (f (-x)) : rfl
... = g (f x)    : by rw hf,
end
```

1.1.6. Imparidad de la composición de funciones impares

Teorema 1.1.10. Sean f y g dos funciones impares, entonces se tiene que la composición de ambas, $(g \circ f)$, es también una función impar.

Demostración: Por la definición 1.1.7 de una función impar vista anteriormente, se tiene que se verifica lo siguiente:

$$f(-x) = -f(x) \qquad \forall x \in \mathbb{R} \text{ y} \tag{1.15}$$

$$g(-x) = -g(x) \qquad \forall x \in \mathbb{R}. \tag{1.16}$$

A continuación, vamos a estudiar la composición de estas dos funciones para un número real x arbitrario:

$$(g \circ f)(-x) = g(f(-x)) \overset{(1)}{=} g(-f(x)) \overset{(2)}{=} -g(f(x)) = -(g \circ f)(x),$$

donde en (1) y en (2) se han usado las hipótesis de funciones impares, (1.15) y (1.16) respectivamente.

De esta forma ya tendríamos la demostración deseada.

□

Prosigamos con la formalización en Lean de esta demostración:

```
import data.real.basic
import algebra.group.pi

def odd_fun (f : ℝ → ℝ) := ∀ x, f (-x) = -f x

example (f g : ℝ → ℝ) : odd_fun f → odd_fun g →  odd_fun (g ∘ f) :=
begin
intros hf hg x,
calc (g ∘ f) (-x)
    = g (f (-x)) : rfl
... = g (-f x)   : by rw hf
... =-g (f x)    : by rw hg
... = -(g ∘ f) x : rfl,
end
```

1.2. Álgebra Básica

En esta segunda sección se estudiarán diversos resultados correspondientes a la asignatura de formación básica: Álgebra Básica. Al igual que Cálculo Infinitesimal, también se corresponde al primer año del grado en Matemáticas.

Se van a estudiar dos resultados para los cuales es necesario introducir previamente el siguiente concepto:

Definición 1.2.1. Denotaremos la división en el conjunto de los números enteros por el símbolo "|". Entonces se dirá que un número entero a divide a otro b si y solamente si existe otro número entero k tal que $b = a \cdot k$. Simbólicamente:

$$a|b \iff \exists k \in \mathbb{Z},\ b = a \cdot k. \tag{1.17}$$

A continuación, una vez hemos introducido el concepto, ya estamos listos para ver los resultados en cuestión.

1.2.1. Transitividad de la división

Teorema 1.2.2. Sean a, b y c tres número enteros tales que verifican que a divide a b y b divide a c, entonces se tiene que a divide a c.

Demostración: Por la definición 1.2.1 de división en el conjunto de los números enteros vista anteriormente, como consecuencia de que a divide a b, se tiene que:

$$\exists k_1 \in \mathbb{Z},\ b = k_1 \cdot a. \tag{1.18}$$

Asimismo, como consecuencia de que b divide a c se verifica que:

$$\exists k_2 \in \mathbb{Z},\ c = k_2 \cdot b. \tag{1.19}$$

De manera que si introducimos la descomposición de b descrita en (1.18) en la de c vista en (1.19), se tiene que:

$$\exists k_1, k_2 \in \mathbb{Z},\ c = \underbrace{k_1 \cdot k_2}_{k_3} \cdot a \implies \exists k_3 \in \mathbb{Z},\ c = k_3 \cdot a. \tag{1.20}$$

Y ya tendríamos el resultado deseado.

□

Veamos ahora la formalización en Lean:

```
import data.real.basic
import data.int.parity

variables (a b c : ℤ)

example (h₁ : a ∣ b) (h₂ : b ∣ c) : a ∣ c :=
begin
cases h₁ with k1 hk1,
cases h₂ with k2 hk2,
rw hk1 at hk2,
use k1*k2,
rw ← mul_assoc,
exact hk2,
end
```

1.2.2. Aditividad de la división

Teorema 1.2.3. Sean a, b y c tres números enteros tales que verifican que a divide a b y a c, entonces se tiene que a divide a la suma de b y c. Simbólicamente:

$$a|b,\ a|c \quad \implies \quad a|(b+c). \tag{1.21}$$

Demostración: Análogamente a la demostración anterior, por la definición 1.2.1, se tiene que:

$$a|b \quad \Longrightarrow \quad \exists k_1 \in \mathbb{Z},\ b = k_1 \cdot a \quad \text{y} \tag{1.22}$$

$$a|c \quad \Longrightarrow \quad \exists k_2 \in \mathbb{Z},\ c = k_2 \cdot a \quad . \tag{1.23}$$

De manera que sumando las dos expresiones descritas en (1.22) y (1.23), se llega a que:

$$\exists k_1, k_2 \in \mathbb{Z},\ b + c = (k_1 + k_2) \cdot a \implies a|(b + c).$$

Teniendo en cuenta la definición 1.2.1, ya se tendría el resultado.

□

La formalización de esta demostración en Lean sería:

```
import data.real.basic
import data.int.parity

variables (a b c : ℤ)

example (h1 : a ∣ b) (h2 : a ∣ c) : a ∣ b+c :=
begin
cases h1 with k1 hk1,
rw hk1,
cases h2 with k2 hk2,
rw hk2,
use k1+k2,
ring,
end
```

Capítulo 2

Problemas de las IMO en Lean

En este capítulo se presentan soluciones de problemas de las Olimpíadas Internacionales de Matemáticas (IMO) formalizadas en Lean.