Calculemus
(Vol. 2: Demostraciones con Lean4)

José A. Alonso Jiménez

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, 10 de julio de 2023 (versión del 19 de agosto de 2023)

Índice general

Capítulo 1

Introducción

Este libro es una recopilación de los ejercicios de demostración con Lean4 que se han ido publicando, desde el 10 de julio de 20023, en el blog Calculemus.

La ordenación de los ejercicios es simplemente temporal según su fecha de publicación en Calculemus y el orden de los ejercicios en Calculemus responde a los que me voy encontrando en mis lecturas.

En cada ejercicio, se comienza proponiendo soluciones en lenguaje natural y, a continuación, se exponen distintas demostraciones con Lean4 ordenadas desde las más detalladas a las más automáticas. Al final de cada ejercicio hay un enlace para interactuar con sus soluciones en Lean4 Web.

Las soluciones del libro están en este repositorio de GitHub.

El libro se irá actualizando periódicamente con los nuevos ejercicios que se proponen diariamente en Calculemus.

Este libro es una continuación de

- DAO (Demostración Asistida por Ordenador) con Lean que es una introducción a la demostración con Lean3 y
- Calculemus (Vol. 1: Demostraciones con Isabelle/HOL y Lean3) que es la recopilación de la primera parte de los ejercicios del blog con demostraciones en Isabelle/HOL y Lean3.

Capítulo 2

Demostraciones de una propiedad de los números enteros

2.1. ∀ m n ∈ ℕ, Even n → Even (m * n)

```
-- ---------------------------------------------------------------------
-- Demostrar que los productos de los números naturales por números
-- pares son pares.
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Si n es par, entonces (por la definición de ‘Even‘) existe un k tal que
--    n = k + k         (1)
-- Por tanto,
--    mn = m(k + k)     (por (1))
--       = mk + mk      (por la propiedad distributiva)
-- Por consiguiente, mn es par.

-- Demostraciones en Lean4
-- =======================

import Mathlib.Data.Nat.Basic
import Mathlib.Data.Nat.Parity
import Mathlib.Tactic

open Nat
```

```
-- 1ª demostración
-- ===============

example : ∀ m n : ℕ, Even n → Even (m * n) := by
  rintro m n ⟨k, hk⟩
  use m * k
  rw [hk]
  ring

-- 2ª demostración
-- ===============

example : ∀ m n : ℕ, Even n → Even (m * n) := by
  rintro m n ⟨k, hk⟩
  use m * k
  rw [hk]
  rw [mul_add]

-- 3ª demostración
-- ===============

example : ∀ m n : ℕ, Even n → Even (m * n) := by
  rintro m n ⟨k, hk⟩
  use m * k
  rw [hk, mul_add]

-- 4ª demostración
-- ===============

example : ∀ m n : Nat, Even n → Even (m * n) := by
  rintro m n ⟨k, hk⟩; use m * k; rw [hk, mul_add]

-- 5ª demostración
-- ===============

example : ∀ m n : ℕ, Even n → Even (m * n) := by
  rintro m n ⟨k, hk⟩
  exact ⟨m * k, by rw [hk, mul_add]⟩

-- 6ª demostración
-- ===============

example : ∀ m n : Nat, Even n → Even (m * n) :=
fun m n ⟨k, hk⟩ ↦ ⟨m * k, by rw [hk, mul_add]⟩
```

```
-- 7ª demostración
-- ===============

example : ∀ m n : ℕ, Even n → Even (m * n) := by
  rintro m n ⟨k, hk⟩
  use m * k
  rw [hk]
  exact mul_add m k k

-- 8ª demostración
-- ===============

example : ∀ m n : ℕ, Even n → Even (m * n) := by
  intros m n hn
  unfold Even at *
  cases hn with
  | intro k hk =>
    use m * k
    rw [hk, mul_add]

-- 9ª demostración
-- ===============

example : ∀ m n : ℕ, Even n → Even (m * n) := by
  intros m n hn
  unfold Even at *
  cases hn with
  | intro k hk =>
    use m * k
    calc m * n
       = m * (k + k)   := by exact congrArg (HMul.hMul m) hk
     _ = m * k + m * k := by exact mul_add m k k

-- 10ª demostración
-- ================

example : ∀ m n : Nat, Even n → Even (m * n) := by
  intros; simp [*, parity_simps]
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

Capítulo 3

Propiedades elementales de los números reales

3.1. En ℝ, (ab)c = b(ac)

```
-- ---------------------------------------------------------------------
-- Demostrar que los números reales tienen la siguiente propiedad
--    (a * b) * c = b * (a * c)
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    (ab)c = (ba)c   [por la conmutativa]
--          = b(ac)   [por la asociativa]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Tactic
import Mathlib.Data.Real.Basic

-- 1ª demostración
example
  (a b c : ℝ)
  : (a * b) * c = b * (a * c) :=
calc
  (a * b) * c = (b * a) * c := by rw [mul_comm a b]
            _ = b * (a * c) := by rw [mul_assoc b a c]
```

```
-- 2ª demostración
example (a b c : ℝ) : (a * b) * c = b * (a * c) :=
by
  rw [mul_comm a b]
  rw [mul_assoc b a c]

-- 3ª demostración
example (a b c : ℝ) : (a * b) * c = b * (a * c) :=
by ring
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.2. En ℝ, (cb)a = b(ac)

```
-- ---------------------------------------------------------------------
-- Demostrar que los números reales tienen la siguiente propiedad
--    (c * b) * a = b * (a * c)
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades:
--    (c * b) * a
--    = (b * c) * a    [por la conmutativa]
--    = b * (c * a)    [por la asociativa]
--    = b * (a * c)    [por la conmutativa]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Tactic
import Mathlib.Data.Real.Basic

-- 1ª demostración
example
  (a b c : ℝ)
  : (c * b) * a = b * (a * c) :=
calc
  (c * b) * a
    = (b * c) * a := by rw [mul_comm c b]
  _ = b * (c * a) := by rw [mul_assoc]
  _ = b * (a * c) := by rw [mul_comm c a]
```

```
-- 2ª demostración
example
  (a b c : ℝ)
  : (c * b) * a = b * (a * c) :=
by
  rw [mul_comm c b]
  rw [mul_assoc]
  rw [mul_comm c a]

-- 3ª demostración
example
  (a b c : ℝ)
  : (c * b) * a = b * (a * c) :=
by ring
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.3. En ℝ, a(bc) = b(ac)

```
-- ---------------------------------------------------------------------
-- Demostrar que los números reales tienen la siguiente propiedad
--    a * (b * c) = b * (a * c)
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades:
--    a(bc)
--    = (ab)c    [por la asociativa]
--    = (ba)c    [por la conmutativa]
--    = b(ac)    [por la asociativa]

-- Demostraciones en Lean4
-- =======================

import Mathlib.Tactic
import Mathlib.Data.Real.Basic

-- 1ª demostración
example
  (a b c : ℝ) : a * (b * c) = b * (a * c) :=
calc
```

```
  a * (b * c)
    = (a * b) * c := by rw [←mul_assoc]
  _ = (b * a) * c := by rw [mul_comm a b]
  _ = b * (a * c) := by rw [mul_assoc]

-- 2ª demostración
example
  (a b c : ℝ) : a * (b * c) = b * (a * c) :=
by
  rw [←mul_assoc]
  rw [mul_comm a b]
  rw [mul_assoc]

-- 3ª demostración
example
  (a b c : ℝ) : a * (b * c) = b * (a * c) :=
by ring
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.4. En ℝ, si ab = cd y e = f, entonces a(be) = c(df)

```
-- ---------------------------------------------------------------------
-- Demostrar que si a, b, c, d, e y f son números reales tales que
--    a * b = c * d y
--    e = f,
-- entonces
--    a * (b * e) = c * (d * f)
-- ---------------------------------------------------------------------

-- Demostración en leguaje natural
-- ===============================

-- Por la siguiente cadena de igualdades
--    a(be)
--    = a(bf)    [por la segunda hipótesis]
--    = (ab)f    [por la asociativa]
--    = (cd)f    [por la primera hipótesis]
--    = c(df)    [por la asociativa]

-- Demostraciones en Lean4
```

```
-- =======================

import Mathlib.Tactic
import Mathlib.Data.Real.Basic

-- 1ª demostración
example
  (a b c d e f : ℝ)
  (h1 : a * b = c * d)
  (h2 : e = f)
  : a * (b * e) = c * (d * f) :=
calc
  a * (b * e)
    = a * (b * f) := by rw [h2]
  _ = (a * b) * f := by rw [←mul_assoc]
  _ = (c * d) * f := by rw [h1]
  _ = c * (d * f) := by rw [mul_assoc]

-- 2ª demostración
example
  (a b c d e f : ℝ)
  (h1 : a * b = c * d)
  (h2 : e = f)
  : a * (b * e) = c * (d * f) :=
by
  rw [h2]
  rw [←mul_assoc]
  rw [h1]
  rw [mul_assoc]

-- 3ª demostración
example
  (a b c d e f : ℝ)
  (h1 : a * b = c * d)
  (h2 : e = f)
  : a * (b * e) = c * (d * f) :=
by
  simp [*, ←mul_assoc]
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.5. En ℝ, si bc = ef, entonces ((ab)c)d = ((ae)f)d

```
-- ---------------------------------------------------------------------
-- Demostrar que si a, b, c, d, e y f son números reales tales que
--    b * c = e * f
-- entonces
--    ((a * b) * c) * d = ((a * e) * f) * d
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    ((ab)c)d
--    = (a(bc))d    [por la asociativa]
--    = (a(ef))d    [por la hipótesis]
--    = ((ae)f)d    [por la asociativa]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Data.Real.Basic
import Mathlib.Tactic

-- 1ª demostración
example
  (a b c d e f : ℝ)
  (h : b * c = e * f)
  : ((a * b) * c) * d = ((a * e) * f) * d :=
calc
  ((a * b) * c) * d
    = (a * (b * c)) * d := by rw [mul_assoc a]
  _ = (a * (e * f)) * d := by rw [h]
  _ = ((a * e) * f) * d := by rw [←mul_assoc a]

-- 2ª demostración
example
  (a b c d e f : ℝ)
  (h : b * c = e * f)
  : ((a * b) * c) * d = ((a * e) * f) * d :=
by
  rw [mul_assoc a]
  rw [h]
  rw [←mul_assoc a]
```

```
-- 3ª demostración
example
  (a b c d e f : ℝ)
  (h : b * c = e * f)
  : ((a * b) * c) * d = ((a * e) * f) * d :=
by
  rw [mul_assoc a, h, ←mul_assoc a]
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.6. En ℝ, si c = ba-d y d = ab, entonces c = 0

```
-- ---------------------------------------------------------------------
-- Demostrar que si a, b, c y d son números reales tales que
--    c = b * a - d
--    d = a * b
-- entonces
--    c = 0
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    c = ba - d     [por la primera hipótesis]
--      = ab - d     [por la conmutativa]
--      = ab - ab    [por la segunda hipótesis]
--      = 0

-- Demostraciones en Lean4
-- =======================

import Mathlib.Data.Real.Basic
import Mathlib.Tactic

-- 1ª demostración
example
  (a b c d : ℝ)
  (h1 : c = b * a - d)
  (h2 : d = a * b)
  : c = 0 :=
calc
```

```
  c = b * a - d     := by rw [h1]
  _ = a * b - d     := by rw [mul_comm]
  _ = a * b - a * b := by rw [h2]
  _ = 0             := by rw [sub_self]

-- 2ª demostración
example
  (a b c d : ℝ)
  (h1 : c = b * a - d)
  (h2 : d = a * b)
  : c = 0 :=
by
  rw [h1]
  rw [mul_comm]
  rw [h2]
  rw [sub_self]

-- 3ª demostración
example
  (a b c d : ℝ)
  (h1 : c = b * a - d)
  (h2 : d = a * b)
  : c = 0 :=
by
  rw [h1, mul_comm, h2, sub_self]
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.7. En ℝ, (a+b)(a+b) = aa+2ab+bb

```
-- ---------------------------------------------------------------------
-- Demostrar que si a y b son números reales, entonces
--    (a + b) * (a + b) = a * a + 2 * (a * b) + b * b
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    (a + b)(a + b)
--    = (a + b)a + (a + b)b    [por la distributiva]
--    = aa + ba + (a + b)b     [por la distributiva]
--    = aa + ba + (ab + bb)    [por la distributiva]
```

```
--     = aa + ba + ab + bb      [por la asociativa]
--     = aa + (ba + ab) + bb    [por la asociativa]
--     = aa + (ab + ab) + bb    [por la conmutativa]
--     = aa + 2(ab) + bb        [por def. de doble]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Data.Real.Basic
import Mathlib.Tactic

variable (a b c : ℝ)

-- 1ª demostración
example :
  (a + b) * (a + b) = a * a + 2 * (a * b) + b * b :=
calc
  (a + b) * (a + b)
    = (a + b) * a + (a + b) * b       := by rw [mul_add]
  _ = a * a + b * a + (a + b) * b     := by rw [add_mul]
  _ = a * a + b * a + (a * b + b * b) := by rw [add_mul]
  _ = a * a + b * a + a * b + b * b   := by rw [←add_assoc]
  _ = a * a + (b * a + a * b) + b * b := by rw [add_assoc (a * a)]
  _ = a * a + (a * b + a * b) + b * b := by rw [mul_comm b a]
  _ = a * a + 2 * (a * b) + b * b     := by rw [←two_mul]

-- 2ª demostración
example :
  (a + b) * (a + b) = a * a + 2 * (a * b) + b * b :=
calc
  (a + b) * (a + b)
    = a * a + b * a + (a * b + b * b) := by rw [mul_add, add_mul, add_mul]
  _ = a * a + (b * a + a * b) + b * b := by rw [←add_assoc, add_assoc (a * a)]
  _ = a * a + 2 * (a * b) + b * b     := by rw [mul_comm b a, ←two_mul]

-- 3ª demostración
example :
  (a + b) * (a + b) = a * a + 2 * (a * b) + b * b :=
calc
  (a + b) * (a + b)
    = a * a + b * a + (a * b + b * b) := by ring
  _ = a * a + (b * a + a * b) + b * b := by ring
  _ = a * a + 2 * (a * b) + b * b     := by ring

-- 4ª demostración
```

```
example :
  (a + b) * (a + b) = a * a + 2 * (a * b) + b * b :=
by ring

-- 5ª demostración
example :
  (a + b) * (a + b) = a * a + 2 * (a * b) + b * b :=
by
  rw [mul_add]
  rw [add_mul]
  rw [add_mul]
  rw [←add_assoc]
  rw [add_assoc (a * a)]
  rw [mul_comm b a]
  rw [←two_mul]

-- 6ª demostración
example :
  (a + b) * (a + b) = a * a + 2 * (a * b) + b * b :=
by
  rw [mul_add, add_mul, add_mul]
  rw [←add_assoc, add_assoc (a * a)]
  rw [mul_comm b a, ←two_mul]

-- 7ª demostración
example :
  (a + b) * (a + b) = a * a + 2 * (a * b) + b * b :=
by linarith
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.8. En ℝ, (a+b)(c+d) = ac+ad+bc+bd

```
-- ---------------------------------------------------------------------
-- Demostrar que si a, b, c y d son números reales, entonces
--    (a + b) * (c + d) = a * c + a * d + b * c + b * d
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    (a + b)(c + d)
```

```
--    = a(c + d) + b(c + d)    [por la distributiva]
--    = ac + ad + b(c + d)     [por la distributiva]
--    = ac + ad + (bc + bd)    [por la distributiva]
--    = ac + ad + bc + bd      [por la asociativa]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Data.Real.Basic
import Mathlib.Tactic

variable (a b c d : ℝ)

-- 1ª demostración
example
  : (a + b) * (c + d) = a * c + a * d + b * c + b * d :=
calc
  (a + b) * (c + d)
    = a * (c + d) + b * (c + d)       := by rw [add_mul]
  _ = a * c + a * d + b * (c + d)     := by rw [mul_add]
  _ = a * c + a * d + (b * c + b * d) := by rw [mul_add]
  _ = a * c + a * d + b * c + b * d   := by rw [←add_assoc]

-- 2ª demostración
example
  : (a + b) * (c + d) = a * c + a * d + b * c + b * d :=
calc
  (a + b) * (c + d)
    = a * (c + d) + b * (c + d)       := by ring
  _ = a * c + a * d + b * (c + d)     := by ring
  _ = a * c + a * d + (b * c + b * d) := by ring
  _ = a * c + a * d + b * c + b * d   := by ring

-- 3ª demostración
example : (a + b) * (c + d) = a * c + a * d + b * c + b * d :=
by ring

-- 4ª demostración
example
  : (a + b) * (c + d) = a * c + a * d + b * c + b * d :=
by
   rw [add_mul]
   rw [mul_add]
   rw [mul_add]
   rw [← add_assoc]
```

```
-- 5ª demostración
example : (a + b) * (c + d) = a * c + a * d + b * c + b * d :=
by rw [add_mul, mul_add, mul_add, ←add_assoc]
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.9. En ℝ, (a+b)(a-b) = a^2-b^2

```
-- ---------------------------------------------------------------------
-- Demostrar que si a y b son números reales, entonces
--    (a + b) * (a - b) = a^2 - b^2
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades:
--    (a + b)(a - b)
--    = a(a - b) + b(a - b)              [por la distributiva]
--    = (aa - ab) + b(a - b)             [por la distributiva]
--    = (a^2 - ab) + b(a - b)            [por def. de cuadrado]
--    = (a^2 - ab) + (ba - bb)           [por la distributiva]
--    = (a^2 - ab) + (ba - b^2)          [por def. de cuadrado]
--    = (a^2 + -(ab)) + (ba - b^2)       [por def. de resta]
--    = a^2 + (-(ab) + (ba - b^2))       [por la asociativa]
--    = a^2 + (-(ab) + (ba + -b^2))      [por def. de resta]
--    = a^2 + ((-(ab) + ba) + -b^2)      [por la asociativa]
--    = a^2 + ((-(ab) + ab) + -b^2)      [por la conmutativa]
--    = a^2 + (0 + -b^2)                 [por def. de opuesto]
--    = (a^2 + 0) + -b^2                 [por asociativa]
--    = a^2 + -b^2                       [por def. de cero]
--    = a^2 - b^2                        [por def. de resta]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Data.Real.Basic
import Mathlib.Tactic

variable (a b : ℝ)

-- 1ª demostración
```

```
-- ===============

example : (a + b) * (a - b) = a^2 - b^2 :=
calc
  (a + b) * (a - b)
    = a * (a - b) + b * (a - b)          := by rw [add_mul]
  _ = (a * a - a * b) + b * (a - b)      := by rw [mul_sub]
  _ = (a^2 - a * b) + b * (a - b)        := by rw [← pow_two]
  _ = (a^2 - a * b) + (b * a - b * b)    := by rw [mul_sub]
  _ = (a^2 - a * b) + (b * a - b^2)      := by rw [← pow_two]
  _ = (a^2 + -(a * b)) + (b * a - b^2)   := by ring
  _ = a^2 + (-(a * b) + (b * a - b^2))   := by rw [add_assoc]
  _ = a^2 + (-(a * b) + (b * a + -b^2))  := by ring
  _ = a^2 + ((-(a * b) + b * a) + -b^2)  := by rw [← add_assoc
                                                (-(a * b)) (b * a) (-b^2)]
  _ = a^2 + ((-(a * b) + a * b) + -b^2)  := by rw [mul_comm]
  _ = a^2 + (0 + -b^2)                   := by rw [neg_add_self (a * b)]
  _ = (a^2 + 0) + -b^2                   := by rw [← add_assoc]
  _ = a^2 + -b^2                         := by rw [add_zero]
  _ = a^2 - b^2                          := by linarith

-- 2ª demostración
-- ===============

example : (a + b) * (a - b) = a^2 - b^2 :=
calc
  (a + b) * (a - b)
    = a * (a - b) + b * (a - b)          := by ring
  _ = (a * a - a * b) + b * (a - b)      := by ring
  _ = (a^2 - a * b) + b * (a - b)        := by ring
  _ = (a^2 - a * b) + (b * a - b * b)    := by ring
  _ = (a^2 - a * b) + (b * a - b^2)      := by ring
  _ = (a^2 + -(a * b)) + (b * a - b^2)   := by ring
  _ = a^2 + (-(a * b) + (b * a - b^2))   := by ring
  _ = a^2 + (-(a * b) + (b * a + -b^2))  := by ring
  _ = a^2 + ((-(a * b) + b * a) + -b^2)  := by ring
  _ = a^2 + ((-(a * b) + a * b) + -b^2)  := by ring
  _ = a^2 + (0 + -b^2)                   := by ring
  _ = (a^2 + 0) + -b^2                   := by ring
  _ = a^2 + -b^2                         := by ring
  _ = a^2 - b^2                          := by ring

-- 3ª demostración
-- ===============
```

```
example : (a + b) * (a - b) = a^2 - b^2 :=
by ring

-- 4ª demostración
-- ===============

-- El lema anterior es
lemma aux : (a + b) * (c + d) = a * c + a * d + b * c + b * d :=
by ring

-- La demostración es
example : (a + b) * (a - b) = a^2 - b^2 :=
by
  rw [sub_eq_add_neg]
  rw [aux]
  rw [mul_neg]
  rw [add_assoc (a * a)]
  rw [mul_comm b a]
  rw [neg_add_self]
  rw [add_zero]
  rw [← pow_two]
  rw [mul_neg]
  rw [← pow_two]
  rw [← sub_eq_add_neg]
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.10. En ℝ, si c = da+b y b = ad, entonces c = 2ad

```
-- ---------------------------------------------------------------------
-- Demostrar que si a, b, c y d son números reales tales que
--    c = d * a + b
--    b = a * d
-- entonces
--    c = 2 * a * d
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
```

```
--    c = da + b     [por la primera hipótesis]
--      = da + ad    [por la segunda hipótesis]
--      = ad + ad    [por la conmutativa]
--      = 2(ad)      [por la def. de doble]
--      = 2ad        [por la asociativa]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Data.Real.Basic
import Mathlib.Tactic

variable (a b c d : ℝ)

-- 1ª demostración
example
  (h1 : c = d * a + b)
  (h2 : b = a * d)
  : c = 2 * a * d :=
calc
  c = d * a + b     := by rw [h1]
  _ = d * a + a * d := by rw [h2]
  _ = a * d + a * d := by rw [mul_comm d a]
  _ = 2 * (a * d)   := by rw [← two_mul (a * d)]
  _ = 2 * a * d     := by rw [mul_assoc]

-- 2ª demostración
example
  (h1 : c = d * a + b)
  (h2 : b = a * d)
  : c = 2 * a * d :=
by
  rw [h2] at h1
  clear h2
  rw [mul_comm d a] at h1
  rw [← two_mul (a*d)] at h1
  rw [← mul_assoc 2 a d] at h1
  exact h1

-- 3ª demostración
example
  (h1 : c = d * a + b)
  (h2 : b = a * d)
  : c = 2 * a * d :=
by rw [h1, h2, mul_comm d a, ← two_mul (a * d), mul_assoc]
```

```
-- 4ª demostración
example
  (h1 : c = d * a + b)
  (h2 : b = a * d)
  : c = 2 * a * d :=
by
  rw [h1]
  rw [h2]
  ring

-- 5ª demostración
example
  (h1 : c = d * a + b)
  (h2 : b = a * d)
  : c = 2 * a * d :=
by
  rw [h1, h2]
  ring

-- 6ª demostración
example
  (h1 : c = d * a + b)
  (h2 : b = a * d)
  : c = 2 * a * d :=
by rw [h1, h2] ; ring

-- 7ª demostración
example
  (h1 : c = d * a + b)
  (h2 : b = a * d)
  : c = 2 * a * d :=
by linarith
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

3.11. En ℝ, si a+b = c, entonces (a+b)(a+b) = ac+bc

```
-- ---------------------------------------------------------------------
-- Demostrar que si a, b y c son números reales tales que
--    a + b = c,
-- entonces
```

```
--    (a + b) * (a + b) = a * c + b * c
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    (a + b)(a + b)
--    = (a + b)c        [por la hipótesis]
--    = ac + bc         [por la distributiva]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Data.Real.Basic
import Mathlib.Tactic

variable (a b c : ℝ)

-- 1ª demostración
example
  (h : a + b = c)
  : (a + b) * (a + b) = a * c + b * c :=
calc
  (a + b) * (a + b)
    = (a + b) * c   := by exact congrArg (HMul.hMul (a + b)) h
  _ = a * c + b * c := by rw [add_mul]

-- 2ª demostración
example
  (h : a + b = c)
  : (a + b) * (a + b) = a * c + b * c :=
by
  nth_rewrite 2 [h]
  rw [add_mul]
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

Capítulo 4

Propiedades elementales de los anillos

4.1. Si R es un anillo y a ∈ R, entonces a + 0 = a

```
-- ---------------------------------------------------------------------
-- Demostrar en Lean4 que si R es un anillo, entonces
--    ∀ a : R, a + 0 = a
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    a + 0 = 0 + a    [por la conmutativa de la suma]
--          = a        [por el axioma del cero por la izquierda]

import Mathlib.Algebra.Ring.Defs

variable {R : Type _} [Ring R]
variable (a : R)

-- Demostraciones con Lean4
-- ========================

-- 1ª demostración
example : a + 0 = a :=
calc a + 0
     = 0 + a := by rw [add_comm]
```

```
  _ = a     := by rw [zero_add]

-- 2ª demostración
example : a + 0 = a :=
by
  rw [add_comm]
  rw [zero_add]

-- 3ª demostración
example : a + 0 = a :=
by rw [add_comm, zero_add]

-- 4ª demostración
example : a + 0 = a :=
by exact add_zero a

-- 5ª demostración
example : a + 0 = a :=
  add_zero a

-- 5ª demostración
example : a + 0 = a :=
by simp
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.2. Si R es un anillo y a ∈ R, entonces a + -a = 0

```
-- ---------------------------------------------------------------------
-- Demostrar en Lean4 que si R es un anillo, entonces
--    ∀ a : R, a + -a = 0
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    a + -a = -a + a    [por la conmutativa de la suma]
--           = 0         [por el axioma de inverso por la izquierda]

import Mathlib.Algebra.Ring.Defs
```

```
variable {R : Type _} [Ring R]
variable (a : R)

-- 1ª demostración
-- ===============

example : a + -a = 0 :=
calc a + -a = -a + a := by rw [add_comm]
          _ = 0      := by rw [add_left_neg]

-- 2ª demostración
-- ===============

example : a + -a = 0 :=
by
  rw [add_comm]
  rw [add_left_neg]

-- 3ª demostración
-- ===============

example : a + -a = 0 :=
by rw [add_comm, add_left_neg]

-- 4ª demostración
-- ===============

example : a + -a = 0 :=
by exact add_neg_self a

-- 5ª demostración
-- ===============

example : a + -a = 0 :=
  add_neg_self a

-- 6ª demostración
-- ===============

example : a + -a = 0 :=
by simp
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.3. Si R es un anillo y a, b ∈ R, entonces -a + (a + b) = b

```
-- ---------------------------------------------------------------------
-- Demostrar en Lean4 que si R es un anillo, entonces
--    ∀ a, b : R, -a + (a + b) = b
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    -a + (a + b) = (-a + a) + b [por la asociativa]
--                 = 0 + b        [por inverso por la izquierda]
--                 = b            [por cero por la izquierda]

import Mathlib.Algebra.Ring.Defs

variable {R : Type _} [Ring R]
variable (a b : R)

-- Demostraciones con Lean4
-- ========================

-- 1ª demostración
example : -a + (a + b) = b :=
calc -a + (a + b) = (-a + a) + b := by rw [← add_assoc]
                _ = 0 + b        := by rw [add_left_neg]
                _ = b            := by rw [zero_add]

-- 2ª demostración
example : -a + (a + b) = b :=
by
  rw [←add_assoc]
  rw [add_left_neg]
  rw [zero_add]

-- 3ª demostración
example : -a + (a + b) = b :=
by rw [←add_assoc, add_left_neg, zero_add]

-- 4ª demostración
example : -a + (a + b) = b :=
by exact neg_add_cancel_left a b
```

```
-- 5ª demostración
example : -a + (a + b) = b :=
  neg_add_cancel_left a b

-- 6ª demostración
example : -a + (a + b) = b :=
by simp
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.4. Si R es un anillo y a, b ∈ R, entonces (a + b) + -b = a

```
-- ---------------------------------------------------------------------
-- Demostrar en Lean4 que si R es un anillo, entonces
--    ∀ a, b : R, (a + b) + -b = a
-- ---------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    (a + b) + -b = a + (b + -b)    [por la asociativa]
--                 _ = a + 0          [por suma con opuesto]
--                 _ = a              [por suma con cero]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Ring.Defs

variable {R : Type _} [Ring R]
variable (a b : R)

-- 1ª demostración
example : (a + b) + -b = a :=
calc
  (a + b) + -b = a + (b + -b) := by rw [add_assoc]
             _ = a + 0        := by rw [add_right_neg]
             _ = a            := by rw [add_zero]

-- 2ª demostración
```

```
example : (a + b) + -b = a :=
by
  rw [add_assoc]
  rw [add_right_neg]
  rw [add_zero]

-- 3ª demostración
example : (a + b) + -b = a :=
by rw [add_assoc, add_right_neg, add_zero]

-- 4ª demostración
example : (a + b) + -b = a :=
  add_neg_cancel_right a b

-- 5ª demostración
example : (a + b) + -b = a :=
  add_neg_cancel_right _ _

-- 6ª demostración
example : (a + b) + -b = a :=
by simp
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.5. Si R es un anillo y a, b, c ∈ R tales que a+b=a+c, entonces b=c

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo y a, b, c ∈ R tales que
--    a + b = a + c
-- entonces
--    b = c
-- ----------------------------------------------------------------------

-- Demostraciones en lenguaje natural (LN)
-- =====================================

-- 1ª demostración en LN
-- =====================

-- Por la siguiente cadena de igualdades
--    b = 0 + b           [por suma con cero]
```

```
--       = (-a + a) + b    [por suma con opuesto]
--       = -a + (a + b)    [por asociativa]
--       = -a + (a + c)    [por hipótesis]
--       = (-a + a) + c    [por asociativa]
--       = 0 + c           [por suma con opuesto]
--       = c               [por suma con cero]

-- 2ª demostración en LN
-- =====================

-- Por la siguiente cadena de implicaciones
--    a + b = a + c
--    ==> -a + (a + b) = -a + (a + c)     [sumando -a]
--    ==>  (-a + a) + b = (-a + a) + c    [por la asociativa]
--    ==>  0 + b = 0 + b                  [suma con opuesto]
--    ==>  b = c                          [suma con cero]

-- 3ª demostración en LN
-- =====================

-- Por la siguiente cadena de igualdades
--    b = -a + (a + b)
--      = -a + (a + c)   [por la hipótesis]
--      = c

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Ring.Defs
import Mathlib.Tactic

variable {R : Type _} [Ring R]
variable {a b c : R}

-- 1ª demostración
example
  (h : a + b = a + c)
  : b = c :=
calc
  b = 0 + b        := by rw [zero_add]
  _ = (-a + a) + b := by rw [add_left_neg]
  _ = -a + (a + b) := by rw [add_assoc]
  _ = -a + (a + c) := by rw [h]
  _ = (-a + a) + c := by rw [←add_assoc]
  _ = 0 + c        := by rw [add_left_neg]
```

```
  _ = c             := by rw [zero_add]

-- 2ª demostración
example
  (h : a + b = a + c)
  : b = c :=
by
  have h1 : -a + (a + b) = -a + (a + c) :=
    congrArg (HAdd.hAdd (-a)) h
  clear h
  rw [← add_assoc] at h1
  rw [add_left_neg] at h1
  rw [zero_add] at h1
  rw [← add_assoc] at h1
  rw [add_left_neg] at h1
  rw [zero_add] at h1
  exact h1

-- 3ª demostración
example
  (h : a + b = a + c)
  : b = c :=
calc
  b = -a + (a + b) := by rw [neg_add_cancel_left a b]
  _ = -a + (a + c) := by rw [h]
  _ = c            := by rw [neg_add_cancel_left]

-- 4ª demostración
example
  (h : a + b = a + c)
  : b = c :=
by
  rw [← neg_add_cancel_left a b]
  rw [h]
  rw [neg_add_cancel_left]

-- 5ª demostración
example
  (h : a + b = a + c)
  : b = c :=
by
  rw [← neg_add_cancel_left a b, h, neg_add_cancel_left]

-- 6ª demostración
example
```

```
  (h : a + b = a + c)
  : b = c :=
add_left_cancel h
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.6. Si R es un anillo y a, b, c ∈ R tales que a+b=c+b, entonces a=c

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo y a, b, c ∈ R tales que
--    a + b = c + b
-- entonces
--    a = c
-- ----------------------------------------------------------------------

-- Demostraciones en lenguaje natural (LN)
-- =======================================

-- 1ª demostración en LN
-- =====================

-- Por la siguiente cadena de igualdades
--    a = a + 0            [por suma con cero]
--      = a + (b + -b)     [por suma con opuesto]
--      = (a + b) + -b     [por asociativa]
--      = (c + b) + -b     [por hipótesis]
--      = c + (b + -b)     [por asociativa]
--      = c + 0            [por suma con opuesto]
--      = c                [por suma con cero]

-- 2ª demostración en LN
-- =====================

-- Por la siguiente cadena de igualdades
--    a = (a + b) + -b
--      = (c + b) + -b     [por hipótesis]
--      = c

-- Demostraciones con Lean4
-- ========================
```

```
import Mathlib.Algebra.Ring.Defs
import Mathlib.Tactic

variable {R : Type _} [Ring R]
variable {a b c : R}

-- 1ª demostración con Lean4
-- =========================

example
  (h : a + b = c + b)
  : a = c :=
calc
  a = a + 0          := by rw [add_zero]
  _ = a + (b + -b) := by rw [add_right_neg]
  _ = (a + b) + -b := by rw [add_assoc]
  _ = (c + b) + -b := by rw [h]
  _ = c + (b + -b) := by rw [← add_assoc]
  _ = c + 0          := by rw [← add_right_neg]
  _ = c              := by rw [add_zero]

-- 2ª demostración con Lean4
-- =========================

example
  (h : a + b = c + b)
  : a = c :=
calc
  a = (a + b) + -b := (add_neg_cancel_right a b).symm
  _ = (c + b) + -b := by rw [h]
  _ = c              := add_neg_cancel_right c b

-- 3ª demostración con Lean4
-- =========================

example
  (h : a + b = c + b)
  : a = c :=
by
  rw [← add_neg_cancel_right a b]
  rw [h]
  rw [add_neg_cancel_right]

-- 4ª demostración con Lean4
-- =========================
```

```
example
  (h : a + b = c + b)
  : a = c :=
by
  rw [← add_neg_cancel_right a b, h, add_neg_cancel_right]

-- 5ª demostración con Lean4
-- =========================

example
  (h : a + b = c + b)
  : a = c :=
add_right_cancel h
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.7. Si R es un anillo y a ∈ R, entonces a.0 = 0

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo y a ∈ R, entonces
--    a * 0 = 0
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Basta aplicar la propiedad cancelativa a
--    a.0 + a.0 = a.0 + 0
-- que se demuestra mediante la siguiente cadena de igualdades
--    a.0 + a.0 = a.(0 + 0)    [por la distributiva]
--              = a.0          [por suma con cero]
--              = a.0 + 0      [por suma con cero]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Ring.Defs
import Mathlib.Tactic

variable {R : Type _} [Ring R]
variable (a : R)
```

```
-- 1ª demostración
-- ===============

example : a * 0 = 0 :=
by
  have h : a * 0 + a * 0 = a * 0 + 0 :=
    calc a * 0 + a * 0 = a * (0 + 0) := by rw [mul_add a 0 0]
                     _ = a * 0       := by rw [add_zero 0]
                     _ = a * 0 + 0   := by rw [add_zero (a * 0)]
  rw [add_left_cancel h]

-- 2ª demostración
-- ===============

example : a * 0 = 0 :=
by
  have h : a * 0 + a * 0 = a * 0 + 0 :=
    calc a * 0 + a * 0 = a * (0 + 0) := by rw [← mul_add]
                     _ = a * 0       := by rw [add_zero]
                     _ = a * 0 + 0   := by rw [add_zero]
  rw [add_left_cancel h]

-- 3ª demostración
-- ===============

example : a * 0 = 0 :=
by
  have h : a * 0 + a * 0 = a * 0 + 0 :=
    by rw [← mul_add, add_zero, add_zero]
  rw [add_left_cancel h]

-- 4ª demostración
-- ===============

example : a * 0 = 0 :=
by
  have : a * 0 + a * 0 = a * 0 + 0 :=
    calc a * 0 + a * 0 = a * (0 + 0) := by simp
                     _ = a * 0       := by simp
                     _ = a * 0 + 0   := by simp
  simp

-- 5ª demostración
-- ===============
```

```
example : a * 0 = 0 :=
  mul_zero a

-- 6ª demostración
-- ===============

example : a * 0 = 0 :=
by simp
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.8. Si R es un anillo y a ∈ R, entonces 0.a = 0

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo y a ∈ R, entonces
--    0 * a = 0
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Basta aplicar la propiedad cancelativa a
--    0.a + 0.a = 0.a + 0
-- que se demuestra mediante la siguiente cadena de igualdades
--    0.a + 0.a = (0 + 0).a    [por la distributiva]
--              = 0.a          [por suma con cero]
--              = 0.a + 0      [por suma con cero]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Ring.Defs
import Mathlib.Tactic

variable {R : Type _} [Ring R]
variable (a : R)

-- 1ª demostración
example : 0 * a = 0 :=
by
  have h : 0 * a + 0 * a = 0 * a + 0 :=
    calc 0 * a + 0 * a = (0 + 0) * a := by rw [add_mul]
                     _ = 0 * a       := by rw [add_zero]
```

```
                    _ = 0 * a + 0   := by rw [add_zero]
  rw [add_left_cancel h]

-- 2ª demostración
example : 0 * a = 0 :=
by
  have h : 0 * a + 0 * a = 0 * a + 0 :=
    by rw [←add_mul, add_zero, add_zero]
  rw [add_left_cancel h]

-- 3ª demostración
example : 0 * a = 0 :=
by
  have : 0 * a + 0 * a = 0 * a + 0 :=
    calc 0 * a + 0 * a = (0 + 0) * a := by simp
                     _ = 0 * a       := by simp
                     _ = 0 * a + 0   := by simp
  simp

-- 4ª demostración
example : 0 * a = 0 :=
by
  have : 0 * a + 0 * a = 0 * a + 0 := by simp
  simp

-- 5ª demostración
example : 0 * a = 0 :=
by simp

-- 6ª demostración
example : 0 * a = 0 :=
zero_mul a
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.9. Si R es un anillo y a, b ∈ R tales que a+b=0, entonces -a=b

```
-- ---------------------------------------------------------------------
-- Demostrar que si es un anillo y a, b ∈ R tales que
--    a + b = 0
-- entonces
```

```
--    -a = b
-- ----------------------------------------------------------------------

-- Demostraciones en lenguaje natural (LN)
-- =======================================

-- 1ª demostración en LN
-- ---------------------

-- Por la siguiente cadena de igualdades
--    -a = -a + 0          [por suma cero]
--       = -a + (a + b)    [por hipótesis]
--       = b               [por cancelativa]

-- 2ª demostración en LN
-- ---------------------

-- Sumando -a a ambos lados de la hipótesis, se tiene
--    -a + (a + b) = -a + 0
-- El término de la izquierda se reduce a b (por la cancelativa) y el de
-- la derecha a -a (por la suma con cero). Por tanto, se tiene
--     b = -a
-- Por la simetría de la igualdad, se tiene
--     -a = b

-- Demostraciones con Lean 4
-- =========================

import Mathlib.Algebra.Ring.Defs
import Mathlib.Tactic

variable {R : Type _} [Ring R]
variable {a b : R}

-- 1ª demostración (basada en la 1º en LN)
example
  (h : a + b = 0)
  : -a = b :=
calc
  -a = -a + 0       := by rw [add_zero]
   _ = -a + (a + b) := by rw [h]
   _ = b            := by rw [neg_add_cancel_left]

-- 2ª demostración (basada en la 1º en LN)
example
```

```
  (h : a + b = 0)
  : -a = b :=
calc
  -a = -a + 0       := by simp
   _ = -a + (a + b) := by rw [h]
   _ = b            := by simp

-- 3ª demostración (basada en la 2º en LN)
example
  (h : a + b = 0)
  : -a = b :=
by
  have h1 : -a + (a + b) = -a + 0 := congrArg (HAdd.hAdd (-a)) h
  have h2 : -a + (a + b) = b := neg_add_cancel_left a b
  have h3 : -a + 0 = -a := add_zero (-a)
  rw [h2, h3] at h1
  exact h1.symm

-- 4ª demostración
example
  (h : a + b = 0)
  : -a = b :=
neg_eq_iff_add_eq_zero.mpr h
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.10. Si R es un anillo y a, b ∈ R tales que a+b=0, entonces a=-b

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo y a, b ∈ R tales que
--    a + b = 0
-- entonces
--    a = -b
-- ----------------------------------------------------------------------

-- Demostraciones en lenguaje natural (LN)
-- ======================================

-- 1ª demostración en LN
-- ---------------------
```

```
-- Por la siguiente cadena de igualdades
--    a = (a + b) + -b    [por la concelativa]
--      = 0 + -b          [por la hipótesis]
--      = -b              [por la suma con cero]

-- 2ª demostración en LN
-- ---------------------

-- Sumando -a a ambos lados de la hipótesis, se tiene
--    (a + b) + -b = 0 + -b
-- El término de la izquierda se reduce a a (por la cancelativa) y el de
-- la derecha a -b (por la suma con cero). Por tanto, se tiene
--     a = -b

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Ring.Defs
import Mathlib.Tactic

variable {R : Type _} [Ring R]
variable {a b : R}

-- 1ª demostración (basada en la 1ª en LN)
example
  (h : a + b = 0)
  : a = -b :=
calc
  a = (a + b) + -b := by rw [add_neg_cancel_right]
  _ = 0 + -b       := by rw [h]
  _ = -b           := by rw [zero_add]

-- 2ª demostración (basada en la 1ª en LN)
example
  (h : a + b = 0)
  : a = -b :=
calc
  a = (a + b) + -b := by simp
  _ = 0 + -b       := by rw [h]
  _ = -b           := by simp

-- 3ª demostración (basada en la 1ª en LN)
example
  (h : a + b = 0)
  : a = -b :=
```

```
by
  have h1 : (a + b) + -b = 0 + -b := by rw [h]
  have h2 : (a + b) + -b = a := add_neg_cancel_right a b
  have h3 : 0 + -b = -b := zero_add (-b)
  rwa [h2, h3] at h1

-- 4ª demostración
example
  (h : a + b = 0)
  : a = -b :=
add_eq_zero_iff_eq_neg.mp h
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.11. Si R es un anillo, entonces -0 = 0

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo, entonces
--    -0 = 0
-- ----------------------------------------------------------------------

-- Demostraciones en lenguaje natural (LN)
-- =======================================

-- 1ª demostración en LN
-- =====================

-- Por la suma con cero se tiene
--    0 + 0 = 0
-- Aplicándole la propiedad
--    ∀ a b ∈ R, a + b = 0 → -a = b
-- se obtiene
--    -0 = 0

-- 2ª demostración en LN
-- =====================

-- Puesto que
--    ∀ a b ∈ R, a + b = 0 → -a = b
-- basta demostrar que
--    0 + 0 = 0
-- que es cierta por la suma con cero.
```

```
-- Demostraciones con Lean4
-- =======================

import Mathlib.Algebra.Ring.Defs
import Mathlib.Tactic

variable {R : Type _} [Ring R]

-- 1ª demostración (basada en la 1ª en LN)
example : (-0 : R) = 0 :=
by
  have h1 : (0 : R) + 0 = 0 := add_zero 0
  show (-0 : R) = 0
  exact neg_eq_of_add_eq_zero_left h1

-- 2ª demostración (basada en la 2ª en LN)
example : (-0 : R) = 0 :=
by
  apply neg_eq_of_add_eq_zero_left
  rw [add_zero]

-- 3ª demostración
example : (-0 : R) = 0 :=
  neg_zero

-- 4ª demostración
example : (-0 : R) = 0 :=
by simp
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.12. Si R es un anillo y a ∈ R, entonces -(-a) = a

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo y a ∈ R, entonces
--     -(-a) = a
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================
```

```
-- Es consecuencia de las siguiente propiedades demostradas en
-- ejercicios anteriores:
--    ∀ a b ∈ R, a + b = 0 → -a = b
--    ∀ a ∈ R, -a + a = 0

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Ring.Defs
import Mathlib.Tactic

variable {R : Type _} [Ring R]
variable {a : R}

-- 1ª demostración
example : -(-a) = a :=
by
  have h1 : -a + a = 0 := add_left_neg a
  show -(-a) = a
  exact neg_eq_of_add_eq_zero_right h1

-- 2ª demostración
example : -(-a) = a :=
by
  apply neg_eq_of_add_eq_zero_right
  rw [add_left_neg]

-- 3ª demostración
example : -(-a) = a :=
neg_neg a

-- 4ª demostración
example : -(-a) = a :=
by simp
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.13. Si R es un anillo y a, b ∈ R, entonces a - b = a + -b

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo y a, b ∈ R, entonces
--    a - b = a + -b
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la definición de la resta.

-- Demostración en Lean4
-- =====================

import Mathlib.Algebra.Ring.Defs

variable {R : Type _} [Ring R]
variable (a b : R)

example : a - b = a + -b :=
-- by exact?
sub_eq_add_neg a b
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.14. Si R es un anillo y a ∈ R, entonces a - a = 0

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo y a ∈ R, entonces
--     a - a = 0
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades:
--    a - a = a + -a    [por definición de resta]
--          = 0         [por suma con opuesto]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Ring.Defs
```

```
variable {R : Type _} [Ring R]
variable (a : R)

-- 1ª demostración
example : a - a = 0 :=
calc
  a - a = a + -a := by rw [sub_eq_add_neg a a]
      _ = 0      := by rw [add_right_neg]

-- 2ª demostración
example : a - a = 0 :=
sub_self a

-- 3ª demostración
example : a - a = 0 :=
by simp
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.15. En los anillos, 1 + 1 = 2

```
-- ---------------------------------------------------------------------
-- Demostrar que en los anillos,
--    1 + 1 = 2
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por cálculo.

-- Demostración con Lean4
-- ======================

import Mathlib.Algebra.Ring.Defs
import Mathlib.Tactic
variable {R : Type _} [Ring R]

-- Demostraciones con Lean4
-- ========================

-- 1ª demostración
example : 1 + 1 = (2 : R) :=
```

```
by norm_num

-- 2ª demostración
example : 1 + 1 = (2 : R) :=
one_add_one_eq_two
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

4.16. Si R es un anillo y a ∈ R, entonces 2a = a+a

```
-- ---------------------------------------------------------------------
-- Demostrar que si R es un anillo y a ∈ R, entonces
--    2 * a = a + a
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    2·a = (1 + 1)·a    [por la definición de 2]
--        = 1·a + 1·a    [por la distributiva]
--        = a + a        [por producto con uno]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Ring.Defs

variable {R : Type _} [Ring R]
variable (a : R)

-- 1ª demostración
example : 2 * a = a + a :=
calc
  2 * a = (1 + 1) * a   := by rw [one_add_one_eq_two]
      _ = 1 * a + 1 * a := by rw [add_mul]
      _ = a + a         := by rw [one_mul]

-- 2ª demostración
example : 2 * a = a + a :=
by exact two_mul a
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

Capítulo 5

Propiedades elementales de los grupos

5.1. Si G es un grupo y $a \in G$, entonces $aa^{-1} = 1$

```
-- ---------------------------------------------------------------------
-- En Lean4, se declara que G es un grupo mediante la expresión
--    variable {G : Type _} [Group G]
--
-- Como consecuencia, se tiene los siguientes axiomas
--    mul_assoc :    ∀ a b c : G, a * b * c = a * (b * c)
--    one_mul :      ∀ a : G, 1 * a = a
--    mul_left_inv : ∀ a : G, a⁻¹ * a = 1
--
-- Demostrar que si G es un grupo y a ∈ G, entonces
--    a * a⁻¹ = 1
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Por la siguiente cadena de igualdades
--    a·a⁻¹ = 1·(a·a⁻¹)                  [por producto con uno]
--          = (1·a)·a⁻¹                  [por asociativa]
--          = (((a⁻¹)⁻¹·a⁻¹) ·a)·a⁻¹    [por producto con inverso]
--          = ((a⁻¹)⁻¹·(a⁻¹ ·a))·a⁻¹    [por asociativa]
--          = ((a⁻¹)⁻¹·1)·a⁻¹            [por producto con inverso]
--          = (a⁻¹)⁻¹·(1·a⁻¹)            [por asociativa]
--          = (a⁻¹)⁻¹·a⁻¹                [por producto con uno]
```

```
--          = 1                         [por producto con inverso]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Group.Defs

variable {G : Type _} [Group G]
variable (a b : G)

-- 1ª demostración
example : a * a⁻¹ = 1 :=
calc
  a * a⁻¹ = 1 * (a * a⁻¹)                  := by rw [one_mul]
        _ = (1 * a) * a⁻¹                  := by rw [mul_assoc]
        _ = (((a⁻¹)⁻¹ * a⁻¹)  * a) * a⁻¹ := by rw [mul_left_inv]
        _ = ((a⁻¹)⁻¹ * (a⁻¹  * a)) * a⁻¹ := by rw [← mul_assoc]
        _ = ((a⁻¹)⁻¹ * 1) * a⁻¹          := by rw [mul_left_inv]
        _ = (a⁻¹)⁻¹ * (1 * a⁻¹)          := by rw [mul_assoc]
        _ = (a⁻¹)⁻¹ * a⁻¹                := by rw [one_mul]
        _ = 1                              := by rw [mul_left_inv]

-- 2ª demostración
example : a * a⁻¹ = 1 :=
calc
  a * a⁻¹ = 1 * (a * a⁻¹)                  := by simp
        _ = (1 * a) * a⁻¹                  := by simp
        _ = (((a⁻¹)⁻¹ * a⁻¹)  * a) * a⁻¹ := by simp
        _ = ((a⁻¹)⁻¹ * (a⁻¹  * a)) * a⁻¹ := by simp
        _ = ((a⁻¹)⁻¹ * 1) * a⁻¹          := by simp
        _ = (a⁻¹)⁻¹ * (1 * a⁻¹)          := by simp
        _ = (a⁻¹)⁻¹ * a⁻¹                := by simp
        _ = 1                              := by simp

-- 3ª demostración
example : a * a⁻¹ = 1 :=
by simp

-- 4ª demostración
example : a * a⁻¹ = 1 :=
by exact mul_inv_self a
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

5.2. Si G es un grupo y a ∈ G, entonces a·1 = a

```
-- ---------------------------------------------------------------------
-- Demostrar que si G es un grupo y a ∈ G, entonces
--    a * 1 = a
-- ----------------------------------------------------------------------

-- Demostración en lenguaje natural
-- ================================

-- Se tiene por la siguiente cadena de igualdades
--    a·1 = a·(a⁻¹·a)    [por producto con inverso]
--        = (a·a⁻¹)·a    [por asociativa]
--        = 1·a          [por producto con inverso]
--        = a            [por producto con uno]

-- Demostraciones con Lean4
-- ========================

import Mathlib.Algebra.Group.Defs

variable {G : Type _} [Group G]
variable (a b : G)

-- 1ª demostración
example : a * 1 = a :=
calc
  a * 1 = a * (a⁻¹ * a) := by rw [mul_left_inv]
      _ = (a * a⁻¹) * a := by rw [mul_assoc]
      _ = 1 * a         := by rw [mul_right_inv]
      _ = a             := by rw [one_mul]

-- 2ª demostración
example : a * 1 = a :=
calc
  a * 1 = a * (a⁻¹ * a) := by simp
      _ = (a * a⁻¹) * a := by simp
      _ = 1 * a         := by simp
      _ = a             := by simp

-- 3ª demostración
example : a * 1 = a :=
by simp
```

```
-- 4ª demostración
example : a * 1 = a :=
by exact mul_one a
```

Se puede interactuar con las pruebas anteriores en Lean 4 Web.

Bibliografía

[1] J. A. Alonso. Lean para matemáticos [1], 2021.

[2] J. A. Alonso. Matemáticas en Lean [2], 2021.

[3] J. A. Alonso. DAO (Demostración Asistida por Ordenador) con Lean [3], 2021.

[4] J. A. Alonso. Calculemus (Vol. 1: Demostraciones con Isabelle/HOL y Lean3) [4] , 2021.

[5] J. Avigad, L. de Moura, and S. Kong. Theorem Proving in Lean4 [5], 2021.

[6] J. Avigad, G. Ebner, and S. Ullrich. The Lean4 Manual [6], 2021.

[7] J. Avigad, M. J. H. Heule, and W. Nawrocki. Logic and mechanized reasoning [7] , 2023.

[8] J. Avigad, R. Y. Lewis, and F. van Doorn. Logic and proof [8] , 2021.

[9] J. Avigad and P. Massot. Mathematics in Lean [9] , 2023.

[10] A. Baanen, A. Bentkamp, J. Blanchette, J. Hölzl, and J. Limperg. The Hitchhiker's Guide to Logical Verification [10], 2020.

[1] https://github.com/jaalonso/Lean_para_matematicos
[2] https://github.com/jaalonso/Matematicas_en_Lean
[3] https://raw.githubusercontent.com/jaalonso/DAO_con_Lean/master/DAO_con_Lean.pdf
[4] https://raw.githubusercontent.com/jaalonso/Calculemus/master/Calculemus.pdf
[5] https://leanprover.github.io/theorem_proving_in_lean4/
[6] https://leanprover.github.io/lean4/doc/
[7] https://avigad.github.io/lamr/logic_and_mechanized_reasoning.pdf
[8] https://leanprover.github.io/logic_and_proof/logic_and_proof.pdf
[9] https://leanprover-community.github.io/mathematics_in_lean/
[10] https://raw.githubusercontent.com/blanchette/logical_verification_2020/master/hitchhikers_guide.pdf

[11] M. Ballard. Transition to advanced mathematics (Thinking and communicating like a mathematician) [11].

[12] K. Buzzard. Sets and logic (in Lean) [12].

[13] K. Buzzard. Functions and relations (in Lean) [13].

[14] K. Buzzard. Course on formalising mathematics [14], 2021.

[15] K. Buzzard. Course on formalising mathematics [15], 2023.

[16] K. Buzzard and M. Pedramfar. The Natural Number Game, version 1.3.3 [16].

[17] D. T. Christiansen. Functional programming in Lean [17], 2023.

[18] M. Dvořák. Lean 4 Cheatsheet [18].

[19] S. Hazratpour. Introduction to proofs [19], 2022.

[20] S. Hazratpour. Introduction to proofs with Lean proof assistant [20], 2022.

[21] R. Lewis. Formal proof and verification, 2022 [21], 2022.

[22] R. Lewis. Discrete structures and probability [22], 2023.

[23] C. Löh. Exploring formalisation (A primer in human-readable mathematics in Lean 3 with examples from simplicial topology) [23], 2022.

[24] H. Macbeth. The mechanics of proof [24], 2023.

[25] P. Massot. Introduction aux mathématiques formalisées [25].

[11] https://300.f22.matthewrobertballard.com/
[12] https://www.ma.imperial.ac.uk/~buzzard/M4000x_html/M40001/M40001_C1.html
[13] https://www.ma.imperial.ac.uk/~buzzard/M4000x_html/M40001/M40001_C2.html
[14] https://github.com/ImperialCollegeLondon/formalising-mathematics
[15] https://github.com/ImperialCollegeLondon/formalising-mathematics-2023
[16] https://www.ma.imperial.ac.uk/~buzzard/xena/natural_number_game/
[17] https://leanprover.github.io/functional_programming_in_lean/
[18] https://raw.githubusercontent.com/madvorak/lean4-cheatsheet/main/lean-tactics.pdf
[19] https://introproofs.github.io/s22/
[20] https://sinhp.github.io/teaching/2022-introduction-to-proofs-with-Lean
[21] https://github.com/BrownCS1951x/fpv2022
[22] https://github.com/brown-cs22/CS22-Lean-2023
[23] https://loeh.app.uni-regensburg.de/mapa/main.pdf
[24] https://hrmacbeth.github.io/math2001/index.html
[25] https://www.imo.universite-paris-saclay.fr/~pmassot/enseignement/math114/

[26] F. L. Roux. Code Lean contenant les preuves d'un cours standard sur les espaces métriques [26], 2020.

[27] W. Schulze. Learning LeanProver [27].

[28] Varios. LFTCM 2020: Lean for the Curious Mathematician 2020 [28].

[29] D. J. Velleman. How to prove it with Lean [29].

[26] https://github.com/FredericLeRoux/LEAN_ESPACES_METRIQUES
[27] https://youtube.com/playlist?list=PLYwF9EIrl42RFQgbmcR_LSCWRIx2WKbXs
[28] https://leanprover-community.github.io/lftcm2020/schedule.html
[29] https://djvelleman.github.io/HTPIwL/