Lógica con Lean

José A. Alonso Jiménez

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, 31 de octubre de 2020

Índice general

Capítulo 1

Introducción

El objetivo de este trabajo es presentar una introducción a la Lógica usando Lean para usarla en las clases de la asignatura de Razonamiento automático del Máster Universitario en Lógica, Computación e Inteligencia Artificial de la Universidad de Sevilla. Por tanto, el único prerrequisito es, como en el Máster, cierta madurez matemática como la que deben tener los alumnos de los Grados de Matemática y de Informática.

El trabajo se basa fundamentalmente en

- El curso de ”Lógica matemática y fundamentos en que se estudia la deducción natural proposicional y de primer orden (basado en el libro Logic in computer science: Modelling and reasoning about systems de Michael Huth y Mark Ryan) y su formalización en Isabelle/HOL.

- Los apuntes de Lógica y demostración con Lean que son un resumen del libro Logic and Proof de Jeremy Avigad, Robert Y. Lewis y Floris van Doorn.

- Los apuntes Deducción natural en Lean en el que se presentan ejemplos de uso de las tácticas de Lean correspondientes a las reglas de la deducción natural.

- Los apuntes Matemáticas en Lean en el que se presentan la formalización en Lean de temas básicos de las matemáticas usando las librerías de mathlib. Está basado en el libro Mathematics in Lean de Jeremy Avigad, Kevin Buzzard, Robert Y. Lewis y Patrick Massot.

La exposición se hará mediante una colección de ejercicios. En cada ejercicios se mostrarán distintas pruebas del mismo resultado y se comentan las tácticas conforme se van usando y los lemas utilizados en las demostraciones.

Además, en cada ejercicio hay tres enlaces: uno al código, otro que al pulsarlo abre el ejercicio en Lean Web (en una sesión del navegador) de forma que se puede navegar por las pruebas y editar otras alternativas, y el tercero es un enlace a un vídeo explicando las soluciones del ejercicio.

El trabajo se desarrolla como un proyecto en GitHub que contiene libro en PDF. Además, los vídeos correspondientes a cada uno de los ejercicios se encuentran en YouTube.

Capítulo 2

Lógica proposicional

2.1. Reglas del condicional

2.1.1. Regla de eliminación del condicional en P → Q, P ⊢ Q

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Eliminación del condicional en Lean
-- ===================================

-- Ej. 1. Demostrar que
--    (P → Q), P ⊢ Q.

import tactic
variables (P Q : Prop)

-- 1ª demostración
example
  (h1 : P → Q)
  (h2 : P)
  : Q :=
begin
  apply h1,
  exact h2,
end

-- 2ª demostración
example
  (h1 : P → Q)
```

```
  (h2 : P)
  : Q :=
begin
  exact h1 h2,
end

-- 3ª demostración
example
  (h1 : P → Q)
  (h2 : P)
  : Q :=
by exact h1 h2

-- 4ª demostración
example
  (h1 : P → Q)
  (h2 : P)
  : Q :=
h1 h2

-- 5ª demostración
example
  (h1 : P → Q)
  (h2 : P)
  : Q :=
by tauto

-- 6ª demostración
example
  (h1 : P → Q)
  (h2 : P)
  : Q :=
by finish

-- 7ª demostración
example
  (h1 : P → Q)
  (h2 : P)
  : Q :=
by solve_by_elim
```

2.1.2. Pruebas de P, P → Q, P → (Q → R) ⊢ R

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de P, P → Q, P → (Q → R) ⊢ R
-- ====================================

-- Ej 1. Demostrar que
--    P, P → Q, P → (Q → R) ⊢ R

import tactic

variables (P Q R : Prop)

-- 1ª demostración
example
  (h1 : P)
  (h2 : P → Q)
  (h3 : P → (Q → R))
  : R :=
have h4 : Q,
  from h2 h1,
have h5 : Q → R,
  from h3 h1,
show R,
  from h5 h4

-- 2ª demostración
example
  (h1 : P)
  (h2 : P → Q)
  (h3 : P → (Q → R))
  : R :=
have h4 : Q     := h2 h1,
have h5 : Q → R := h3 h1,
show R, from h5 h4

-- 3ª demostración
example
  (h1 : P)
  (h2 : P → Q)
  (h3 : P → (Q → R))
  : R :=
show R, from (h3 h1) (h2 h1)
```

```
-- 4ª demostración
example
  (h1 : P)
  (h2 : P → Q)
  (h3 : P → (Q → R))
  : R :=
(h3 h1) (h2 h1)

-- 5ª demostración
example
  (h1 : P)
  (h2 : P → Q)
  (h3 : P → (Q → R))
  : R :=
by finish
```

2.1.3. Regla de introducción del condicional en P → P

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Introducción del condicional en Lean
-- ====================================

-- Ej. 1. Demostrar que
--    P → P

import tactic
variable (P : Prop)

-- 1ª demostración
example : P → P :=
assume h : P,
show P, from h

-- 2ª demostración
example : P → P :=
assume : P,
show P, from this

-- 3ª demostración
example : P → P :=
assume : P,
show P, from ‹P›
```

```
-- 4ª demostración
example : P → P :=
assume h : P, h

-- 5ª demostración
example : P → P :=
λ h, h

-- 6ª demostración
example : P → P :=
id

-- 7ª demostración
example : P → P :=
begin
  intro h,
  exact h,
end

-- 8ª demostración
example : P → P :=
begin
  intro,
  exact ‹P›,
end

-- 9ª demostración
example : P → P :=
begin
  intro h,
  assumption,
end

-- 10ª demostración
example : P → P :=
begin
  intro,
  assumption,
end

-- 11ª demostración
example : P → P :=
by tauto
```

```
-- 12ª demostración
example : P → P :=
by finish

-- 13ª demostración
example : P → P :=
by simp
```

2.1.4. Pruebas de P → (Q → P)

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de P → (Q → P)
-- ======================

-- Ej. 1. Demostrar
--    P → (Q → P)

import tactic

variables (P Q : Prop)

-- 1ª demostración
example : P → (Q → P) :=
assume (h1 : P),
show Q → P, from
  ( assume h2 : Q,
    show P, from h1)

-- 2ª demostración
example : P → (Q → P) :=
assume (h1 : P),
show Q → P, from
  ( assume h2 : Q, h1)

-- 3ª demostración
example : P → (Q → P) :=
assume (h1 : P),
show Q → P, from
  ( λ h2, h1)

-- 4ª demostración
example : P → (Q → P) :=
```

```
assume (h1 : P), (λ h2, h1)

-- 5ª demostración
example : P → (Q → P) :=
λ h1, λ h2, h1

-- 6ª demostración
example : P → (Q → P) :=
λ h1 h2, h1

-- 7ª demostración
example : P → (Q → P) :=
λ h _, h

-- 8ª demostración
example : P → (Q → P) :=
imp_intro

-- 9ª demostración
example : P → (Q → P) :=
begin
  intro h1,
  intro h2,
  exact h1,
end

-- 10ª demostración
example : P → (Q → P) :=
begin
  intros h1 h2,
  exact h1,
end

-- 6ª demostración
example : P → (Q → P) :=
λ h1 h2, h1

-- 11ª demostración
example : P → (Q → P) :=
by tauto

-- 12ª demostración
example : P → (Q → P) :=
by finish
```

2.1.5. Pruebas del silogismo hipotético: $P \to Q, Q \to R \vdash P \to R$

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas del silogismo hipotético
-- --------------------------------

import tactic

variables (P Q R : Prop)

-- Ej. 1. Demostrar que
--    P → Q, Q → R ⊢ P → R

-- 1º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
assume h : P,
have h3 : Q,
  from h1 h,
show R,
  from h2 h3

-- 2º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
assume h : P,
have h3 : Q := h1 h,
show R,
  from h2 h3

-- 3º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
assume h : P,
show R,
  from h2 (h1 h)
```

```
-- 4º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
assume h : P, h2 (h1 h)

-- 5º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
λ h, h2 (h1 h)

-- 6º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
h2 ∘ h1

-- 7º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
begin
  intro h,
  apply h2,
  apply h1,
  exact h,
end

-- 8º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
begin
  intro h,
  apply h2,
  exact h1 h,
end
```

```
-- 9º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
begin
  intro h,
  exact h2 (h1 h),
end

-- 10º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
λ h, h2 (h1 h)

-- 11º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
h2 ∘ h1

-- 12º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
by tauto

-- 13º demostración
example
  (h1 : P → Q)
  (h2 : Q → R)
  : P → R :=
by finish
```

2.2. Reglas de la conjunción

2.2.1. Reglas de la conjunción en P ∧ Q, R ⊢ Q ∧ R

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Reglas de la conjunción
-- =======================

-- Ej. 1. Demostrar que
--    P ∧ Q, R ⊢ Q ∧ R

import tactic

variables (P Q R : Prop)

-- 1ª demostración
-- ===============

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
have hQ : Q,
  from and.right hPQ,
show Q ∧ R,
  from and.intro hQ hR

-- 2ª demostración
-- ===============

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
have hQ : Q,
  from hPQ.right,
show Q ∧ R,
  from ⟨hQ, hR⟩

-- 3ª demostración
-- ===============

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
have hQ : Q,
  from hPQ.2,
show Q ∧ R,
  from ⟨hQ, hR⟩
```

```
-- 4ª demostración
-- ===============

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
have hQ : Q :=
  hPQ.2,
show Q ∧ R,
  from ⟨hQ, hR⟩

-- 5ª demostración
-- ===============

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
show Q ∧ R,
  from ⟨hPQ.2, hR⟩

-- 6ª demostración
-- ===============

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
⟨hPQ.2, hR⟩

-- 7ª demostración
-- ===============

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
match hPQ with ⟨hP, hQ⟩ :=
  and.intro hQ hR
end

-- 8ª demostración
-- ===============
```

```
example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
begin
  split,
  { cases hPQ with hP hQ,
    clear hP,
    exact hQ, },
  { exact hR, },
end

-- 9ª demostración
-- ===============

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
begin
  split,
  { cases hPQ,
    assumption, },
  { assumption, },
end

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

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
begin
  constructor,
  { cases hPQ,
    assumption, },
  { assumption, },
end

-- 11ª demostración
-- ===============

example
```

```
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
by tauto

-- 12ª demostración
-- ===============

example
  (hPQ : P ∧ Q)
  (hR : R)
  : Q ∧ R :=
by finish
```

2.2.2. Pruebas de P ∧ Q → Q ∧ P

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas en Lean de P ∧ Q → Q ∧ P
-- ================================

-- Ej. 1. Demostrar que
--    P ∧ Q → Q ∧ P

import tactic

variables (P Q : Prop)

-- 1ª demostración
-- ===============

example : P ∧ Q → Q ∧ P :=
assume h : P ∧ Q,
have hP : P,
  from and.left h,
have hQ : Q,
  from and.right h,
show Q ∧ P,
  from and.intro hQ hP

-- 2ª demostración
-- ===============
```

```
example : P ∧ Q → Q ∧ P :=
assume h : P ∧ Q,
have hP : P,
  from h.left,
have hQ : Q,
  from h.right,
show Q ∧ P,
  from ⟨hQ, hP⟩

-- 3ª demostración
-- ===============

example : P ∧ Q → Q ∧ P :=
assume h : P ∧ Q,
have hP : P,
  from h.1,
have hQ : Q,
  from h.2,
show Q ∧ P,
  from ⟨hQ, hP⟩

-- 4ª demostración
-- ===============

example : P ∧ Q → Q ∧ P :=
assume h : P ∧ Q,
have hP : P := h.1,
have hQ : Q := h.2,
show Q ∧ P,
  from ⟨hQ, hP⟩

-- 5ª demostración
-- ===============

example : P ∧ Q → Q ∧ P :=
assume h : P ∧ Q,
show Q ∧ P,
  from ⟨h.2, h.1⟩

-- 6ª demostración
-- ===============

example : P ∧ Q → Q ∧ P :=
assume h : P ∧ Q, ⟨h.2, h.1⟩
```

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

example : P ∧ Q → Q ∧ P :=
λ h, ⟨h.2, h.1⟩

-- 8ª demostración
-- ===============

example : P ∧ Q → Q ∧ P :=
and.comm.mp

-- 9ª demostración
-- ===============

example : P ∧ Q → Q ∧ P :=
begin
  intro h,
  cases h with hP hQ,
  split,
  { exact hQ, },
  { exact hP, },
end

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

example : P ∧ Q → Q ∧ P :=
begin
  rintro ⟨hP, hQ⟩,
  exact ⟨hQ, hP⟩,
end

-- 11ª demostración
-- ===============

example : P ∧ Q → Q ∧ P :=
λ ⟨hP, hQ⟩, ⟨hQ, hP⟩

-- 12ª demostración
-- ===============

example : P ∧ Q → Q ∧ P:=
by tauto
```

```
-- 13ª demostración
-- ===============

example : P ∧ Q → Q ∧ P :=
by finish
```

2.3. Reglas de la negación

2.3.1. Reglas de la negación con (⊥ ⊢ P), (P, ¬P ⊢ ⊥) y ¬(P ∧ ¬P)

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Reglas de la negación
-- =====================

import tactic

variable (P : Prop)

-- Eliminación del falso
-- ---------------------

-- Ej. 1. Demostrar que
--    ⊥ ⊢ P

-- 1ª demostración
example
  (h : false)
  : P :=
false.elim h

-- 2ª demostración
example
  (h : false)
  : P :=
false.rec P h

-- 3ª demostración
example
  (h : false)
```

```
  : P :=
by tauto

-- 4ª demostración
example
  (h : false)
  : P :=
by cases h

-- 5ª demostración
example
  (h : false)
  : P :=
by finish

-- 6ª demostración
example
  (h : false)
  : P :=
by solve_by_elim

-- Definición de la negación
-- -------------------------

-- ¬P ≡ (P → false)

-- Eliminación de la negación
-- --------------------------

-- Ej. 2. Demostrar que
--    P, ¬P ⊢ ⊥

-- 1ª demostración
example
  (h1: P)
  (h2: ¬P)
  : false :=
not.elim h2 h1

-- 2ª demostración
example
  (h1: P)
  (h2: ¬P)
  : false :=
h2 h1
```

```
-- Introducción de la negación
-- ---------------------------

-- Ej. 3. Demostrar
--    ¬(P ∧ ¬P)

-- 1ª demostración
example : ¬(P ∧ ¬P) :=
not.intro
  ( assume h : P ∧ ¬P,
    have h1 : P  := h.1,
    have h2 : ¬P := h.2,
    show false, from h2 h1 )

-- 2ª demostración
example : ¬(P ∧ ¬P) :=
not.intro
  ( assume h : P ∧ ¬P,
    show false, from h.2 h.1 )

-- 3ª demostración
example : ¬(P ∧ ¬P) :=
not.intro
  ( assume h : P ∧ ¬P, h.2 h.1 )

-- 4ª demostración
example : ¬(P ∧ ¬P) :=
not.intro (λ h, h.2 h.1)

-- 5ª demostración
example : ¬(P ∧ ¬P) :=
begin
  intro h,
  cases h with h1 h2,
  apply h2,
  exact h1,
end

-- 6ª demostración
example : ¬(P ∧ ¬P) :=
begin
  rintro ⟨h1, h2⟩,
  exact h2 h1,
end
```

```
-- 7ª demostración
example : ¬(P ∧ ¬P) :=
λ ⟨h1, h2⟩, h2 h1

-- 8ª demostración
example : ¬(P ∧ ¬P) :=
(and_not_self P).mp

-- 9ª demostración
example : ¬(P ∧ ¬P) :=
by tauto

-- 10ª demostración
example : ¬(P ∧ ¬P) :=
by finish

-- 11ª demostración
example : ¬(P ∧ ¬P) :=
by simp
```

2.3.2. Pruebas de P → Q, P → ¬Q ⊢ ¬P

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de P → Q, P → ¬Q ⊢ ¬P
-- =============================

import tactic

variables (P Q : Prop)

-- Ej. 1. Demostrar
--    P → Q, P → ¬Q ⊢ ¬P

-- 1ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
assume h : P,
have h4 : Q,
  from h1 h,
```

```
have h5 : ¬Q,
  from h2 h,
show false,
  from h5 h4

-- 2ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
assume h : P,
have h4 : Q  := h1 h,
have h5 : ¬Q := h2 h,
show false,
  from h5 h4

-- 3ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
assume h : P,
show false,
  from (h2 h) (h1 h)

-- 4ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
assume h : P, (h2 h) (h1 h)

-- 5ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
λ h, (h2 h) (h1 h)

-- 6ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
begin
```

```
  intro h,
  have h3 : ¬Q := h2 h,
  apply h3,
  apply h1,
  exact h,
end

-- 7ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
begin
  intro h,
  have h3 : ¬Q := h2 h,
  apply h3,
  exact h1 h,
end

-- 8ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
begin
  intro h,
  have h3 : ¬Q := h2 h,
  exact h3 (h1 h),
end

-- 9ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
begin
  intro h,
  exact (h2 h) (h1 h),
end

-- 10ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
```

```
λ h, (h2 h) (h1 h)

-- 11ª demostración
example
  (h1 : P → Q)
  (h2 : P → ¬Q)
  : ¬P :=
by finish
```

2.3.3. Pruebas del modus tollens: P → Q, ¬Q ⊢ ¬P

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas del modus tollens
-- =========================

-- Ej. 1. Demostrar
--    P → Q, ¬Q ⊢ ¬P

import tactic

variables (P Q : Prop)

-- 1ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
assume h3 : P,
have h4 : Q,
  from h1 h3,
show false,
  from h2 h4

-- 2ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
assume h3 : P,
have h4 : Q := h1 h3,
show false,
  from h2 h4
```

```
-- 3ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
assume h3 : P,
show false,
  from h2 (h1 h3)

-- 4ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
assume h3 : P, h2 (h1 h3)

-- 5ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
λ h, h2 (h1 h)

-- 6ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
h2 ∘ h1

-- 7ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
mt h1 h2

-- 8ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
by tauto
```

```
-- 9ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
by finish

-- 10ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
begin
  intro h,
  apply h2,
  apply h1,
  exact h,
end

-- 11ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
begin
  intro h,
  exact h2 (h1 h),
end

-- 12ª demostración
example
  (h1 : P → Q)
  (h2 : ¬Q)
  : ¬P :=
λ h, h2 (h1 h)
```

2.3.4. Pruebas de P → (Q → R), P, ¬R ⊢ ¬Q

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de P → (Q → R), P, ¬R ⊢ ¬Q
-- ==================================
```

```
-- Ej. 1. Demostrar
--    Pruebas de P → (Q → R), P, ¬R ⊢ ¬Q

import tactic

variables (P Q R : Prop)

-- 1ª demostración
example
  (h1 : P → (Q → R))
  (h2 : P)
  (h3 : ¬R)
  : ¬Q :=
have h4 : Q → R,
  from h1 h2,
show ¬Q,
  from mt h4 h3

-- 2ª demostración
example
  (h1 : P → (Q → R))
  (h2 : P)
  (h3 : ¬R)
  : ¬Q :=
have h4 : Q → R := h1 h2,
show ¬Q,
  from mt h4 h3

-- 3ª demostración
example
  (h1 : P → (Q → R))
  (h2 : P)
  (h3 : ¬R)
  : ¬Q :=
show ¬Q,
  from mt (h1 h2) h3

-- 4ª demostración
example
  (h1 : P → (Q → R))
  (h2 : P)
  (h3 : ¬R)
  : ¬Q :=
mt (h1 h2) h3
```

```
-- 5ª demostración
example
  (h1 : P → (Q → R))
  (h2 : P)
  (h3 : ¬R)
  : ¬Q :=
begin
  intro h4,
  apply h3,
  apply (h1 h2),
  exact h4,
end

-- 6ª demostración
example
  (h1 : P → (Q → R))
  (h2 : P)
  (h3 : ¬R)
  : ¬Q :=
begin
  intro h4,
  apply h3,
  exact (h1 h2) h4,
end

-- 7ª demostración
example
  (h1 : P → (Q → R))
  (h2 : P)
  (h3 : ¬R)
  : ¬Q :=
begin
  intro h4,
  exact h3 ((h1 h2) h4),
end

-- 8ª demostración
example
  (h1 : P → (Q → R))
  (h2 : P)
  (h3 : ¬R)
  : ¬Q :=
λ h4, h3 ((h1 h2) h4)

-- 9ª demostración
```

```
example
  (h1 : P → (Q → R))
  (h2 : P)
  (h3 : ¬R)
  : ¬Q :=
by finish
```

2.3.5. Pruebas de P → Q ⊢ ¬Q → ¬P

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de P → Q ⊢ ¬Q → ¬P
-- ==========================

-- Ej. 1. Demostrar
--    P → Q ⊢ ¬Q → ¬P

import tactic

variables (P Q : Prop)

-- 1ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
assume h2 : ¬Q,
show ¬P,
  from mt h1 h2

-- 2ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
assume h2 : ¬Q, mt h1 h2

-- 3ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
λ h2, mt h1 h2

-- 4ª demostración
example
```

```
  (h1 : P → Q)
  : ¬Q → ¬P :=
mt h1

-- 5ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
begin
  intro h2,
  exact mt h1 h2,
end

-- 6ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
begin
  intro h2,
  intro h3,
  apply h2,
  apply h1,
  exact h3,
end

-- 7ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
begin
  intro h2,
  intro h3,
  apply h2,
  exact h1 h3,
end

-- 8ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
begin
  intro h2,
  intro h3,
  exact h2 (h1 h3),
end
```

```
-- 9ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
begin
  intros h2 h3,
  exact h2 (h1 h3),
end

-- 10ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
λ h2 h3, h2 (h1 h3)

-- 11ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
by tauto

-- 12ª demostración
example
  (h1 : P → Q)
  : ¬Q → ¬P :=
by finish
```

2.3.6. Regla de introducción de la doble negación: P ⊢ ¬¬P

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de introducción de la doble negación
-- ==========================================

-- Ej. 1. Demostrar
--    P ⊢ ¬¬P

import tactic
variable (P : Prop)

-- 1ª demostración
```

```
example
  (h1 : P)
  : ¬¬P :=
not.intro
  ( assume h2: ¬P,
    show false,
      from h2 h1)

-- 2ª demostración
example
  (h1 : P)
  : ¬¬P :=
assume h2: ¬P,
show false,
  from h2 h1

-- 3ª demostración
example
  (h1 : P)
  : ¬¬P :=
assume h2: ¬P, h2 h1

-- 4ª demostración
example
  (h1 : P)
  : ¬¬P :=
λ h2, h2 h1

-- 5ª demostración
example
  (h1 : P)
  : ¬¬P :=
not_not.mpr h1

-- 6ª demostración
example
  (h1 : P)
  : ¬¬P :=
not_not_intro h1

-- 7ª demostración
example
  (h1 : P)
  : ¬¬P :=
begin
```

```
  intro h2,
  exact h2 h1,
end

-- 8ª demostración
example
  (h1 : P)
  : ¬¬P :=
by tauto

-- 9ª demostración
example
  (h1 : P)
  : ¬¬P :=
by finish
```

2.3.7. Pruebas de ¬Q → ¬P ⊢ P → ¬¬Q

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de ¬Q → ¬P ⊢ P → ¬¬Q
-- ============================

-- Ej. 1. Demostrar
--    ¬Q → ¬P ⊢ P → ¬¬Q

import tactic

variables (P Q : Prop)

-- 1ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
assume h2 : P,
have h3 : ¬¬P,
  from not_not_intro h2,
show ¬¬Q,
  from mt h1 h3

-- 2ª demostración
example
  (h1 : ¬Q → ¬P)
```

```
  : P → ¬¬Q :=
assume h2 : P,
have h3 : ¬¬P := not_not_intro h2,
show ¬¬Q,
  from mt h1 h3

-- 3ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
assume h2 : P,
show ¬¬Q,
  from mt h1 (not_not_intro h2)

-- 4ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
assume h2 : P, mt h1 (not_not_intro h2)

-- 5ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
λ h2, mt h1 (not_not_intro h2)

-- 6ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
imp_not_comm.mp h1

-- 7ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
begin
  intro h2,
  apply mt h1,
  apply not_not_intro,
  exact h2,
end

-- 8ª demostración
example
```

```
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
begin
  intro h2,
  apply mt h1,
  exact not_not_intro h2,
end

-- 9ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
begin
  intro h2,
  exact mt h1 (not_not_intro h2),
end

-- 10ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
λ h2, mt h1 (not_not_intro h2)

-- 11ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
begin
  intro h2,
  intro h3,
  have h4 : ¬P := h1 h3,
  exact h4 h2,
end

-- 12ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
begin
  intros h2 h3,
  exact (h1 h3) h2,
end

-- 13ª demostración
example
```

```
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
λ h2 h3, (h1 h3) h2

-- 14ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
by tauto

-- 15ª demostración
example
  (h1 : ¬Q → ¬P)
  : P → ¬¬Q :=
by finish
```

2.4. Reglas de la disyunción

2.4.1. Reglas de introducción de la disyunción

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Reglas de introducción de la disyunción
-- =======================================

import tactic

variables (P Q R : Prop)

-- Ej. 1. Demostrar
--    P ⊢ P ∨ Q

-- 1ª demostración
example
  (h : P)
  : P ∨ Q :=
or.intro_left Q h

-- 2ª demostración
example
  (h : P)
  : P ∨ Q :=
```

```
or.inl h

-- 3ª demostración
example
  (h : P)
  : P ∨ Q :=
by tauto

-- 4ª demostración
example
  (h : P)
  : P ∨ Q :=
by finish

-- Ej. 2. Demostrar
--    P ∧ Q ⊢ P ∨ R

-- 1ª demostración
example
  (h1 : P ∧ Q)
  : P ∨ R :=
have h2 : P,
  from and.elim_left h1,
show P ∨ R,
  from or.inl h2

-- 2ª demostración
example
  (h1 : P ∧ Q)
  : P ∨ R :=
have h2 : P,
  from h1.1,
show P ∨ R,
  from or.inl h2

-- 3ª demostración
example
  (h1 : P ∧ Q)
  : P ∨ R :=
have h2 : P := h1.1,
show P ∨ R,
  from or.inl h2

-- 4ª demostración
example
```

```
  (h1 : P ∧ Q)
  : P ∨ R :=
show P ∨ R,
  from or.inl h1.1

-- 5ª demostración
example
  (h1 : P ∧ Q)
  : P ∨ R :=
or.inl h1.1

-- 6ª demostración
example
  (h1 : P ∧ Q)
  : P ∨ R :=
by tauto

-- 7ª demostración
example
  (h1 : P ∧ Q)
  : P ∨ R :=
by finish

-- Ej. 3. Demostrar
--    Q ⊢ P ∨ Q

-- 1ª demostración
example
  (h : Q)
  : P ∨ Q :=
or.intro_right P h

-- 2ª demostración
example
  (h : Q)
  : P ∨ Q :=
or.inr h

-- 3ª demostración
example
  (h : Q)
  : P ∨ Q :=
by tauto

-- 4ª demostración
```

```
example
  (h : Q)
  : P ∨ Q :=
by finish

-- Ej. 4. Demostrar
--    P ∧ Q ⊢ P ∨ R

-- 1ª demostración
example
  (h1 : P ∧ Q)
  : R ∨ Q :=
have h2 : Q,
  from and.elim_right h1,
show R ∨ Q,
  from or.inr h2

-- 2ª demostración
example
  (h1 : P ∧ Q)
  : R ∨ Q :=
have h2 : Q,
  from h1.2,
show R ∨ Q,
  from or.inr h2

-- 3ª demostración
example
  (h1 : P ∧ Q)
  : R ∨ Q :=
have h2 : Q := h1.2,
show R ∨ Q,
  from or.inr h2

-- 4ª demostración
example
  (h1 : P ∧ Q)
  : R ∨ Q :=
show R ∨ Q,
  from or.inr h1.2

-- 5ª demostración
example
  (h1 : P ∧ Q)
  : R ∨ Q :=
```

```
or.inr h1.2

-- 6ª demostración
example
  (h1 : P ∧ Q)
  : R ∨ Q :=
by tauto

-- 7ª demostración
example
  (h1 : P ∧ Q)
  : R ∨ Q :=
by finish
```

2.4.2. Regla de eliminación de la disyunción

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de eliminación de la disyunción
-- =====================================

import tactic

variables (P Q R : Prop)

-- Ej. 1. Demostrar
--    P ∨ Q, P → R, Q → R ⊢ R

-- 1ª demostración
example
  (h1 : P ∨ Q)
  (h2 : P → R)
  (h3 : Q → R)
  : R :=
or.elim h1 h2 h3

-- 2ª demostración
example
  (h1 : P ∨ Q)
  (h2 : P → R)
  (h3 : Q → R)
  : R :=
h1.elim h2 h3
```

```
-- 3ª demostración
example
  (h1 : P ∨ Q)
  (h2 : P → R)
  (h3 : Q → R)
  : R :=
or.rec h2 h3 h1

-- 4ª demostración
example
  (h1 : P ∨ Q)
  (h2 : P → R)
  (h3 : Q → R)
  : R :=
begin
  cases h1 with hP hQ,
  { exact h2 hP, },
  { exact h3 hQ, },
end

-- 5ª demostración
example
  (h1 : P ∨ Q)
  (h2 : P → R)
  (h3 : Q → R)
  : R :=
by tauto

-- 6ª demostración
example
  (h1 : P ∨ Q)
  (h2 : P → R)
  (h3 : Q → R)
  : R :=
by finish
```

2.4.3. Pruebas de P ∨ Q ⊢ Q ∨ P

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de P ∨ Q ⊢ Q ∨ P
-- ========================
```

```
import tactic

variables (P Q R : Prop)

-- Ej. 1. Demostrar
--    P ∨ Q ⊢ Q ∨ P

-- 1ª demostración
example
  (h1 : P ∨ Q)
  : Q ∨ P :=
or.elim h1
  ( assume h2 : P,
    show Q ∨ P,
      from or.inr h2 )
  ( assume h3 : Q,
    show Q ∨ P,
      from or.inl h3 )

-- 2ª demostración
example
  (h1 : P ∨ Q)
  : Q ∨ P :=
or.elim h1
  ( λ h, or.inr h )
  ( λ h, or.inl h )

-- 3ª demostración
example
  (h1 : P ∨ Q)
  : Q ∨ P :=
or.elim h1 or.inr or.inl

example
  (h1 : P ∨ Q)
  : Q ∨ P :=
h1.elim or.inr or.inl

-- 4ª demostración
example
  (h1 : P ∨ Q)
  : Q ∨ P :=
or.rec or.inr or.inl h1
```

```
-- 5ª demostración
example
  (h1 : P ∨ Q)
  : Q ∨ P :=
or.swap h1

-- 6ª demostración
example
  (h1 : P ∨ Q)
  : Q ∨ P :=
begin
  cases h1 with h2 h3,
  { exact or.inr h2, },
  { exact or.inl h3, },
end

-- 7ª demostración
example
  (P ∨ Q)
  : Q ∨ P :=
begin
  cases ‹P ∨ Q›,
  { exact or.inr ‹P›, },
  { exact or.inl ‹Q›, },
end

-- 8ª demostración
example
  (h1 : P ∨ Q)
  : Q ∨ P :=
begin
  cases h1 with h2 h3,
  { right,
    exact h2, },
  { left,
    exact h3, },
end

-- 9ª demostración
example
  (h1 : P ∨ Q)
  : Q ∨ P :=
by tauto
```

```
-- 10ª demostración
example
  (h1 : P ∨ Q)
  : Q ∨ P :=
by finish
```

2.4.4. Pruebas de Q → R ⊢ P ∨ Q → P ∨ R

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de Q → R ⊢ P ∨ Q → P ∨ R
-- ================================

-- Ej. 1. Demostrar
--    Q → R ⊢ P ∨ Q → P ∨ R

import tactic

variables (P Q R : Prop)

-- 1ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
assume h2 : P ∨ Q,
or.elim h2
  ( assume h3 : P,
    show P ∨ R,
      from or.inl h3 )
  ( assume h4 : Q,
    have h5 : R := h1 h4,
    show P ∨ R,
      from or.inr h5 )

-- 2ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
assume h2 : P ∨ Q,
or.elim h2
  ( assume h3 : P, or.inl h3 )
  ( assume h4 : Q,
    show P ∨ R,
```

```
      from or.inr (h1 h4) )

-- 3ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
assume h2 : P ∨ Q,
or.elim h2
  ( assume h3 : P, or.inl h3 )
  ( assume h4 : Q, or.inr (h1 h4) )

-- 4ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
assume h2 : P ∨ Q,
or.elim h2
  ( λ h3, or.inl h3 )
  ( λ h4, or.inr (h1 h4) )

-- 5ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
assume h2 : P ∨ Q,
or.elim h2
  or.inl
  ( λ h, or.inr (h1 h) )

-- 6ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
λ h2, or.elim h2 or.inl (λ h, or.inr (h1 h))

example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
λ h2, h2.elim or.inl (λ h, or.inr (h1 h))

-- 7ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
or.imp_right h1
```

```
-- 8ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
begin
  intro h2,
  cases h2 with h3 h4,
  { exact or.inl h3, },
  { exact or.inr (h1 h4), },
end

-- 9ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
begin
  intro h2,
  cases h2 with h3 h4,
  { left,
    exact h3, },
  { right,
    exact (h1 h4), },
end

-- 10ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
begin
  rintro (h3 | h4),
  { left,
    exact h3, },
  { right,
    exact (h1 h4), },
end

-- 11ª demostración
example
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
by tauto

-- 12ª demostración
example
```

```
  (h1 : Q → R)
  : P ∨ Q → P ∨ R :=
by finish
```

2.4.5. Pruebas de ¬P ∨ Q ⊢ P → Q

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Prueba de ¬P ∨ Q ⊢ P → Q
-- ========================

-- Ej. 1. Demostrar
--    ¬P ∨ Q ⊢ P → Q

import tactic

variables (P Q : Prop)

-- 1ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
assume h2 : P,
or.elim h1
  ( assume h3 : ¬P,
    have h4 : false,
      from h3 h2,
    show Q,
      from false.elim h4)
  ( assume h5 : Q,
    show Q, from h5)

-- 2ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
assume h2 : P,
or.elim h1
  ( assume h3 : ¬P,
    have h4 : false,
      from h3 h2,
    show Q,
      from false.elim h4)
```

```
  ( assume h5 : Q, h5)

-- 3ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
assume h2 : P,
or.elim h1
  ( assume h3 : ¬P,
    have h4 : false,
      from h3 h2,
    show Q,
      from false.elim h4)
  ( λ h5, h5)

-- 4ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
assume h2 : P,
or.elim h1
  ( assume h3 : ¬P,
    have h4 : false,
      from h3 h2,
    show Q,
      from false.elim h4)
  id

-- 5ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
assume h2 : P,
or.elim h1
  ( assume h3 : ¬P,
    show Q,
      from false.elim (h3 h2))
  id

-- 6ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
assume h2 : P,
or.elim h1
```

```
  ( assume h3 : ¬P, false.elim (h3 h2))
  id

-- 7ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
assume h2 : P,
or.elim h1
  ( λ h3, false.elim (h3 h2))
  id

-- 8ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
λ h2, or.elim h1 (λ h3, false.elim (h3 h2)) id

example
  (h1 : ¬P ∨ Q)
  : P → Q :=
λ h2, h1.elim (λ h3, false.elim (h3 h2)) id

example
  (h1 : ¬P ∨ Q)
  : P → Q :=
λ h2, h1.elim (λ h3, (h3 h2).elim) id

-- 9ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
imp_iff_not_or.mpr h1

-- 10ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
begin
  intro h2,
  cases h1 with h3 h4,
  { apply false.rec,
    exact h3 h2, },
  { exact h4, },
end
```

```
-- 11ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
begin
  intro h2,
  cases h1 with h3 h4,
  { exact false.elim (h3 h2), },
  { exact h4, },
end

-- 12ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
begin
  intro h2,
  cases h1 with h3 h4,
  { exfalso,
    exact h3 h2, },
  { exact h4, },
end

-- 13ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
by tauto

-- 14ª demostración
example
  (h1 : ¬P ∨ Q)
  : P → Q :=
by finish
```

2.5. Reglas del bicondicional

2.5.1. Regla de introducción del bicondicional en P ∧ Q ↔ Q ∧ P

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de introducción del bicondicional
-- ========================================

-- Ej. 1. Demostrar
--     P ∧ Q ↔ Q ∧ P

import tactic

variables (P Q : Prop)

-- 1ª demostración
example : P ∧ Q ↔ Q ∧ P :=
iff.intro
  ( assume h1 : P ∧ Q,
    have h2 : P,
      from and.elim_left h1,
    have h3 : Q,
      from and.elim_right h1,
    show Q ∧ P,
      from and.intro h3 h2)
  ( assume h4 : Q ∧ P,
    have h5 : Q,
      from and.elim_left h4,
    have h6 : P,
      from and.elim_right h4,
    show P ∧ Q,
      from and.intro h6 h5)

-- 2ª demostración
example : P ∧ Q ↔ Q ∧ P :=
iff.intro
  ( assume h1 : P ∧ Q,
    have h2 : P,
      from h1.1,
    have h3 : Q,
      from h1.2,
    show Q ∧ P,
      from and.intro h3 h2)
  ( assume h4 : Q ∧ P,
    have h5 : Q,
      from h4.1,
    have h6 : P,
      from h4.2,
    show P ∧ Q,
      from and.intro h6 h5)
```

```
-- 3ª demostración
example : P ∧ Q ↔ Q ∧ P :=
iff.intro
  ( assume h1 : P ∧ Q,
    have h2 : P := h1.1,
    have h3 : Q := h1.2,
    show Q ∧ P,
      from and.intro h3 h2)
  ( assume h4 : Q ∧ P,
    have h5 : Q := h4.1,
    have h6 : P := h4.2,
    show P ∧ Q,
      from and.intro h6 h5)

-- 4ª demostración
example : P ∧ Q ↔ Q ∧ P :=
iff.intro
  ( assume h1 : P ∧ Q,
    show Q ∧ P,
      from and.intro h1.2 h1.1)
  ( assume h4 : Q ∧ P,
    show P ∧ Q,
      from and.intro h4.2 h4.1)

-- 5ª demostración
example : P ∧ Q ↔ Q ∧ P :=
iff.intro
  ( assume h1 : P ∧ Q, and.intro h1.2 h1.1)
  ( assume h4 : Q ∧ P, and.intro h4.2 h4.1)

-- 6ª demostración
example : P ∧ Q ↔ Q ∧ P :=
iff.intro
  ( assume h1 : P ∧ Q, ⟨h1.2, h1.1⟩)
  ( assume h4 : Q ∧ P, ⟨h4.2, h4.1⟩)

-- 7ª demostración
example : P ∧ Q ↔ Q ∧ P :=
iff.intro
  ( λ h, ⟨h.2, h.1⟩)
  ( λ h, ⟨h.2, h.1⟩)

-- 8ª demostración
lemma aux :
```

```
  P ∧ Q → Q ∧ P :=
λ h, ⟨h.2, h.1⟩

example : P ∧ Q ↔ Q ∧ P :=
iff.intro (aux P Q) (aux Q P)

-- 9ª demostración
example : P ∧ Q ↔ Q ∧ P :=
and.comm

-- 10ª demostración
example : P ∧ Q ↔ Q ∧ P :=
begin
  split,
  { intro h1,
    cases h1 with h2 h3,
    split,
    { exact h3, },
    { exact h2, }},
  { intro h4,
    cases h4 with h5 h6,
    split,
    { exact h6, },
    { exact h5, }},
end

-- 11ª demostración
example : P ∧ Q ↔ Q ∧ P :=
begin
  split,
  { rintro ⟨h2, h3⟩,
    split,
    { exact h3, },
    { exact h2, }},
  { rintro ⟨h5, h6⟩,
    split,
    { exact h6, },
    { exact h5, }},
end

-- 12ª demostración
example : P ∧ Q ↔ Q ∧ P :=
begin
  constructor,
  { rintro ⟨h2, h3⟩,
```

```
    constructor,
    { exact h3, },
    { exact h2, }},
  { rintro ⟨h5, h6⟩,
    constructor,
    { exact h6, },
    { exact h5, }},
end

-- 13ª demostración
example : P ∧ Q ↔ Q ∧ P :=
by tauto

-- 14ª demostración
example : P ∧ Q ↔ Q ∧ P :=
by finish
```

2.5.2. Reglas de eliminación del bicondicional en P ↔ Q, P ∨ Q ⊢ P ∧ Q

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Reglas de eliminacion del condicional
-- =====================================

-- Ej. 1. Demostrar
--     P ↔ Q, P ∨ Q ⊢ P ∧ Q

import tactic

variables (P Q : Prop)

-- 1ª demostración
example
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
or.elim h2
  ( assume h3 : P,
    have h4 : P → Q,
      from iff.elim_left h1,
    have h5 : Q,
```

```
      from h4 h3,
    show P ∧ Q,
      from and.intro h3 h5 )
  ( assume h6 : Q,
    have h7 : Q → P,
      from iff.elim_right h1,
    have h8 : P,
      from h7 h6,
    show P ∧ Q,
      from and.intro h8 h6 )

-- 2ª demostración
example
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
or.elim h2
  ( assume h3 : P,
    have h4 : P → Q := h1.1,
    have h5 : Q := h4 h3,
    show P ∧ Q, from ⟨h3, h5⟩ )
  ( assume h6 : Q,
    have h7 : Q → P := h1.2,
    have h8 : P := h7 h6,
    show P ∧ Q, from ⟨h8, h6⟩ )

-- 3ª demostración
example
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
or.elim h2
  ( assume h3 : P,
    show P ∧ Q, from ⟨h3, (h1.1 h3)⟩ )
  ( assume h6 : Q,
    show P ∧ Q, from ⟨h1.2 h6, h6⟩ )

-- 4ª demostración
example
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
or.elim h2
  (λh, ⟨h, (h1.1 h)⟩)
  (λh, ⟨h1.2 h, h⟩)
```

```
example
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
h2.elim
  (λh, ⟨h, (h1.1 h)⟩)
  (λh, ⟨h1.2 h, h⟩)

-- 5ª demostración
example
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
begin
  cases h2 with h3 h4,
  { split,
    { exact h3, },
    { apply h1.mp,
      exact h3, }},
  { split,
    { apply h1.mpr,
      exact h4, },
    { exact h4, }},
end

-- 6ª demostración
example
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
begin
  cases h2 with h3 h4,
  { split,
    { exact h3, },
    { rw ← h1,
      exact h3, }},
  { split,
    { rw h1,
      exact h4, },
    { exact h4, }},
end

-- 7ª demostración
example
```

```
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
by tauto

-- 8ª demostración
example
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
by finish

-- 9ª demostración
example
  (h1 : P ↔ Q)
  (h2 : P ∨ Q)
  : P ∧ Q :=
begin
  simp [h1] at h2 |-,
  assumption,
end
```

2.6. Reglas de la lógica clásica

2.6.1. Pruebas de la regla de reducción al absurdo

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Prueba de la regla de reducción al absurdo
-- ==========================================

import tactic

variable (P : Prop)

-- 1ª demostración
example
  (h1 : ¬P → false)
  : P :=
have h2 : ¬¬P, from
  assume h3 : ¬P,
  show false, from h1 h3,
```

```
show P, from not_not.mp h2

-- 2ª demostración
example
  (h1 : ¬P → false)
  : P :=
begin
  apply not_not.mp,
  intro h2,
  exact h1 h2,
end

-- 3ª demostración
example
  (h1 : ¬P → false)
  : P :=
begin
  apply not_not.mp,
  exact λ h2, h1 h2,
end

-- 4ª demostración
example
  (h1 : ¬P → false)
  : P :=
not_not.mp (λ h2, h1 h2)

#print axioms not_not

-- 5ª demostración
example
  (h1 : ¬P → false)
  : P :=
by_contra h1

#print axioms by_contra

-- 6ª demostración
lemma RAA
  (h1 : ¬P → false)
  : P :=
by finish

#print axioms RAA
```

2.6.2. Pruebas de la eliminación de la doble negación

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de la eliminación de la doble negación
-- ==============================================

-- Ej. 1. Demostrar
--      ¬¬P ⊢ P

import tactic

variable (P : Prop)

open_locale classical

-- 1ª demostración
example
  (h1 : ¬¬P)
  : P :=
by_contra
  ( assume h2 : ¬P,
    show false,
      from h1 h2 )

-- 2ª demostración
example
  (h1 : ¬¬P)
  : P :=
by_contra
  ( assume h2 : ¬P,
    h1 h2 )

-- 3ª demostración
example
  (h1 : ¬¬P)
  : P :=
by_contra (λ h2, h1 h2)

-- 4ª demostración
example
  (h1 : ¬¬P)
  : P :=
not_not.mp h1
```

```
-- 5ª demostración
example
  (h1 : ¬¬P)
  : P :=
begin
  by_contradiction h2,
  exact h1 h2,
end

-- 6ª demostración
example
  (h1 : ¬¬P)
  : P :=
by tauto

-- 7ª demostración
lemma aux
  (h1 : ¬¬P)
  : P :=
by finish

#print axioms aux
```

2.6.3. Pruebas del principio del tercio excluso

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas del principio del tercio excluso
-- ========================================

-- Demostrar que
--    F ∨ ¬F

import tactic

variable (F : Prop)

open_locale classical

-- 1ª demostración
example : F ∨ ¬F :=
by_contradiction
  ( assume h1 : ¬(F ∨ ¬F),
```

```
    have h2 : ¬F, from
      assume h3 : F,
      have h4 : F ∨ ¬F, from or.inl h3,
      show false, from h1 h4,
    have h5 : F ∨ ¬F, from or.inr h2,
    show false, from h1 h5 )

-- 2ª demostración
example : F ∨ ¬F :=
by_contradiction
  ( assume h1 : ¬(F ∨ ¬F),
    have h2 : ¬F, from
      assume h3 : F,
      have h4 : F ∨ ¬F, from or.inl h3,
      show false, from h1 h4,
    have h5 : F ∨ ¬F, from or.inr h2,
    h1 h5 )

-- 3ª demostración
example : F ∨ ¬F :=
by_contradiction
  ( assume h1 : ¬(F ∨ ¬F),
    have h2 : ¬F, from
      assume h3 : F,
      have h4 : F ∨ ¬F, from or.inl h3,
      show false, from h1 h4,
    h1 (or.inr h2) )

-- 4ª demostración
example : F ∨ ¬F :=
by_contradiction
  ( assume h1 : ¬(F ∨ ¬F),
    have h2 : ¬F, from
      assume h3 : F,
      have h4 : F ∨ ¬F, from or.inl h3,
      h1 h4,
    h1 (or.inr h2) )

-- 5ª demostración
example : F ∨ ¬F :=
by_contradiction
  ( assume h1 : ¬(F ∨ ¬F),
    have h2 : ¬F, from
      assume h3 : F,
      h1 (or.inl h3),
```

```
    h1 (or.inr h2) )

-- 6ª demostración
example : F ∨ ¬F :=
by_contradiction
  ( assume h1 : ¬(F ∨ ¬F),
    have h2 : ¬F, from
      λ h3, h1 (or.inl h3),
    h1 (or.inr h2) )

-- 7ª demostración
example : F ∨ ¬F :=
by_contradiction
  ( assume h1 : ¬(F ∨ ¬F),
    h1 (or.inr (λ h3, h1 (or.inl h3))) )

-- 8ª demostración
example : F ∨ ¬F :=
by_contradiction
  ( λ h1, h1 (or.inr (λ h3, h1 (or.inl h3))) )

-- 9ª demostración
example : F ∨ ¬F :=
em F

#print axioms em

-- 10ª demostración
example : F ∨ ¬F :=
begin
  by_contra h1,
  apply h1,
  apply or.inr,
  intro h2,
  apply h1,
  exact or.inl h2,
end

-- 11ª demostración
example : F ∨ ¬F :=
begin
  by_contra h1,
  apply h1,
  apply or.inr,
  intro h2,
```

```
  exact h1 (or.inl h2),
end

-- 12ª demostración
example : F ∨ ¬F :=
begin
  by_contra h1,
  apply h1,
  apply or.inr,
  exact λ h2, h1 (or.inl h2),
end

-- 13ª demostración
example : F ∨ ¬F :=
begin
  by_contra h1,
  apply h1,
  exact or.inr (λ h2, h1 (or.inl h2)),
end

-- 14ª demostración
example : F ∨ ¬F :=
begin
  by_contra h1,
  exact h1 (or.inr (λ h2, h1 (or.inl h2))),
end

-- 15ª demostración
example : F ∨ ¬F :=
by_contra (λ h1, h1 (or.inr (λh2, h1 (or.inl h2))))

-- 16ª demostración
example : F ∨ ¬F :=
begin
  by_contra h1,
  apply h1,
  right,
  intro h2,
  apply h1,
  left,
  exact h2,
end

-- 17ª demostración
example : F ∨ ¬F :=
```

```
by tauto

-- 18ª demostración
example : F ∨ ¬F :=
by finish
```

2.6.4. Pruebas de P → Q ⊢ ¬P ∨ Q

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de P → Q ⊢ ¬P ∨ Q
-- =========================

-- Ej. 1. Demostrar
--    P → Q ⊢ ¬P ∨ Q

import tactic

variables (P Q : Prop)

open_locale classical

-- 1ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
have h2 : P ∨ ¬P,
  from em P,
or.elim h2
  ( assume h3 : P,
    have h4 : Q,
      from h1 h3,
    show ¬P ∨ Q,
      from or.inr h4)
  ( assume h5 : ¬P,
    show ¬P ∨ Q,
      from or.inl h5)

-- 2ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
have h2 : P ∨ ¬P,
```

```
  from em P,
or.elim h2
  ( assume h3 : P,
    have h4 : Q,
      from h1 h3,
    show ¬P ∨ Q,
      from or.inr h4)
  ( assume h5 : ¬P,
    or.inl h5)

-- 3ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
have h2 : P ∨ ¬P,
  from em P,
or.elim h2
  ( assume h3 : P,
    have h4 : Q,
      from h1 h3,
    show ¬P ∨ Q,
      from or.inr h4)
  ( λ h5, or.inl h5)

-- 4ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
have h2 : P ∨ ¬P,
  from em P,
or.elim h2
  ( assume h3 : P,
    have h4 : Q,
      from h1 h3,
    or.inr h4)
  ( λ h5, or.inl h5)

-- 5ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
have h2 : P ∨ ¬P,
  from em P,
or.elim h2
  ( assume h3 : P,
```

```
    or.inr (h1 h3))
  ( λ h5, or.inl h5)

-- 6ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
have h2 : P ∨ ¬P,
  from em P,
or.elim h2
  ( λ h3, or.inr (h1 h3))
  ( λ h5, or.inl h5)

-- 7ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
or.elim (em P)
  ( λ h3, or.inr (h1 h3))
  ( λ h5, or.inl h5)

example
  (h1 : P → Q)
  : ¬P ∨ Q :=
(em P).elim
  ( λ h3, or.inr (h1 h3))
  ( λ h5, or.inl h5)

-- 8ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
-- by library_search
not_or_of_imp h1

-- 9ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
if h3 : P then or.inr (h1 h3) else or.inl h3

-- 10ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
```

```
begin
  by_cases h2 : P,
  { apply or.inr,
    exact h1 h2, },
  { exact or.inl h2, },
end

-- 11ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
begin
  by_cases h2 : P,
  { exact or.inr (h1 h2), },
  { exact or.inl h2, },
end

-- 12ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
begin
  by_cases h2 : P,
  { right,
    exact h1 h2, },
  { left,
    exact h2, },
end

-- 13ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
-- by hint
by tauto

-- 14ª demostración
example
  (h1 : P → Q)
  : ¬P ∨ Q :=
-- by hint
by finish
```

2.6.5. Pruebas de P, ¬¬(Q ∧ R) ⊢ ¬¬P ∧ R

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de P, ¬¬(Q ∧ R) ⊢ ¬¬P ∧ R
-- =================================

-- Ej. 1. Demostrar
--    P, ¬¬(Q ∧ R) ⊢ ¬¬P ∧ R

import tactic

variables (P Q R : Prop)

open_locale classical

-- 1ª demostración
example
  (h1 : P)
  (h2 : ¬¬(Q ∧ R))
  : ¬¬P ∧ R :=
have h3 : ¬¬P,   from not_not_intro h1,
have h4 : Q ∧ R, from not_not.mp h2,
have h5 : R,     from and.elim_right h4,
show ¬¬P ∧ R,    from and.intro h3 h5

-- 2ª demostración
example
  (h1 : P)
  (h2 : ¬¬(Q ∧ R))
  : ¬¬P ∧ R :=
have h3 : ¬¬P,   from not_not_intro h1,
have h4 : Q ∧ R, from not_not.mp h2,
have h5 : R,     from and.elim_right h4,
and.intro h3 h5

-- 3ª demostración
example
  (h1 : P)
  (h2 : ¬¬(Q ∧ R))
  : ¬¬P ∧ R :=
have h3 : ¬¬P,   from not_not_intro h1,
have h4 : Q ∧ R, from not_not.mp h2,
have h5 : R,     from h4.2,
and.intro h3 h5
```

```
-- 5ª demostración
example
  (h1 : P)
  (h2 : ¬¬(Q ∧ R))
  : ¬¬P ∧ R :=
and.intro (not_not_intro h1) (not_not.mp h2).2

-- 6ª demostración
example
  (h1 : P)
  (h2 : ¬¬(Q ∧ R))
  : ¬¬P ∧ R :=
begin
  split,
  { exact not_not_intro h1, },
  { push_neg at h2,
    exact h2.2, },
end

-- 7ª demostración
example
  (h1 : P)
  (h2 : ¬¬(Q ∧ R))
  : ¬¬P ∧ R :=
-- by hint
by tauto

-- 8ª demostración
lemma aux
  (h1 : P)
  (h2 : ¬¬(Q ∧ R))
  : ¬¬P ∧ R :=
by finish

#print axioms aux
```

2.6.6. Pruebas de ¬P → Q, ¬Q ⊢ P

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de ¬P → Q, ¬Q ⊢ P
-- =========================
```

```
-- Ej. 1. Demostrar
--     ¬P → Q, ¬Q ⊢ P

import tactic

variables (P Q : Prop)

open_locale classical

-- 1ª demostración
example
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
have h3 : ¬¬P, from mt h1 h2,
show P,        from not_not.mp h3

-- 2ª demostración
example
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
not_not.mp (mt h1 h2)

-- 3ª demostración
example
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
begin
  by_contra h3,
  apply h2,
  exact h1 h3,
end

-- 4ª demostración
example
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
begin
  by_contra h3,
  exact h2 (h1 h3),
end
```

```
-- 5ª demostración
example
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
by_contra (λ h3, h2 (h1 h3))

-- 6ª demostración
example
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
by_contra (λ h3, (h2 ∘ h1) h3)

-- 7ª demostración
example
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
by_contra (h2 ∘ h1)

-- 8ª demostración
example
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
-- by library_search
not_not.mp (mt h1 h2)

-- 9ª demostración
example
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
-- by hint
by tauto

-- 10ª demostración
lemma aux
  (h1 : ¬P → Q)
  (h2 : ¬Q)
  : P :=
-- by hint
by finish
```

```
#print axioms aux
```

2.6.7. Pruebas de (Q → R) → ((¬Q → ¬P) → (P → R))

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de (Q → R) → ((¬Q → ¬P) → (P → R))
-- ===========================================

-- Ej. 1. Demostrar
--    (Q → R) → ((¬Q → ¬P) → (P → R))

import tactic

variables (P Q R : Prop)

-- 1ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
assume h1 : Q → R,
show (¬Q → ¬P) → (P → R), from
  ( assume h2 : ¬Q → ¬P,
    show P → R, from
      ( assume h3 : P,
        have h4 : ¬¬P, from not_not_intro h3,
        have h5 : ¬¬Q, from mt h2 h4,
        have h6 : Q,   from not_not.mp h5,
        show R, from h1 h6))

-- 2ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
assume h1 : Q → R,
show (¬Q → ¬P) → (P → R), from
  ( assume h2 : ¬Q → ¬P,
    show P → R, from
      ( assume h3 : P,
        have h4 : ¬¬P, from not_not_intro h3,
        have h5 : ¬¬Q, from mt h2 h4,
        have h6 : Q,   from not_not.mp h5,
        h1 h6))
```

```
-- 3ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
assume h1 : Q → R,
show (¬Q → ¬P) → (P → R), from
  ( assume h2 : ¬Q → ¬P,
    show P → R, from
      ( assume h3 : P,
        have h4 : ¬¬P, from not_not_intro h3,
        have h5 : ¬¬Q, from mt h2 h4,
        h1 (not_not.mp h5)))

-- 4ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
assume h1 : Q → R,
show (¬Q → ¬P) → (P → R), from
  ( assume h2 : ¬Q → ¬P,
    show P → R, from
      ( assume h3 : P,
        have h4 : ¬¬P, from not_not_intro h3,
        h1 (not_not.mp (mt h2 h4))))

-- 5ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
assume h1 : Q → R,
show (¬Q → ¬P) → (P → R), from
  ( assume h2 : ¬Q → ¬P,
    show P → R, from
      ( assume h3 : P,
        h1 (not_not.mp (mt h2 (not_not_intro h3)))))

-- 6ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
assume h1 : Q → R,
show (¬Q → ¬P) → (P → R), from
  ( assume h2 : ¬Q → ¬P,
    show P → R, from
      (λh3, h1 (not_not.mp (mt h2 (not_not_intro h3)))))

-- 7ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
```

```
assume h1 : Q → R,
show (¬Q → ¬P) → (P → R), from
  ( assume h2 : ¬Q → ¬P,
    (λh3, h1 (not_not.mp (mt h2 (not_not_intro h3)))))

-- 8ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
assume h1 : Q → R,
show (¬Q → ¬P) → (P → R), from
  ( λh2,
    (λh3, h1 (not_not.mp (mt h2 (not_not_intro h3)))))

-- 9ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
assume h1 : Q → R,
(λ h2 h3, h1 (not_not.mp (mt h2 (not_not_intro h3))))

-- 10ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
λ h1 h2 h3, h1 (not_not.mp (mt h2 (not_not_intro h3)))

-- 11ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
begin
  intro h1,
  intro h2,
  intro h3,
  apply h1,
  apply not_not.mp,
  apply mt h2,
  exact not_not_intro h3,
end

-- 12ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
begin
  intros h1 h2 h3,
  apply h1,
  apply not_not.mp,
  apply mt h2,
```

```
  exact not_not_intro h3,
end

-- 13ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
begin
  intros h1 h2 h3,
  apply h1,
  apply not_not.mp,
  exact mt h2 (not_not_intro h3),
end

-- 14ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
begin
  intros h1 h2 h3,
  exact h1 (not_not.mp (mt h2 (not_not_intro h3))),
end

-- 15ª demostración
example :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
λ h1 h2 h3, h1 (not_not.mp (mt h2 (not_not_intro h3)))

-- 16ª demostración
lemma aux :
  (Q → R) → ((¬Q → ¬P) → (P → R)) :=
-- by hint
by finish

#print axioms aux
```

Capítulo 3

Lógica de primer orden

3.1. Reglas del cuantificador universal

3.1.1. Regla de eliminación del cuantificador universal

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de eliminación del cuantificador universal
-- ===============================================

-- Ej. 1. Demostrar
--    P(c), ∀x (P(x) → ¬Q(x)) ⊢ ¬Q(c)

import tactic

variable  U : Type
variable  c : U
variables P Q : U → Prop

-- 1ª demostración
example
  (h1 : P c)
  (h2 : ∀x, P x → ¬Q x)
  : ¬Q c :=
have h3 : P c → ¬Q c, from h2 c,
show ¬Q c,            from h3 h1

-- 2ª demostración
example
  (h1 : P c)
  (h2 : ∀x, P x → ¬Q x)
```

```
  : ¬Q c :=
have h3 : P c → ¬Q c, from h2 c,
h3 h1

-- 3ª demostración
example
  (h1 : P c)
  (h2 : ∀x, P x → ¬Q x)
  : ¬Q c :=
(h2 c) h1

-- 4ª demostración
example
  (h1 : P c)
  (h2 : ∀x, P x → ¬Q x)
  : ¬Q c :=
-- by library_search
h2 c h1

-- 5ª demostración
example
  (h1 : P c)
  (h2 : ∀x, P x → ¬Q x)
  : ¬Q c :=
-- by hint
by tauto

-- 6ª demostración
example
  (h1 : P c)
  (h2 : ∀x, P x → ¬Q x)
  : ¬Q c :=
by finish

-- 7ª demostración
example
  (h1 : P c)
  (h2 : ∀x, P x → ¬Q x)
  : ¬Q c :=
begin
  apply h2,
  exact h1,
end
```

3.1.2. Regla de introducción del cuantificador universal

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de introducción del cuantificador universal
-- =================================================

-- Ej. 1. Demostrar
--    ∀x [P(x) → ¬Q(x)], ∀x P(x) ⊢ ∀x ¬Q(x)

import tactic

variable  U : Type
variables P Q : U → Prop

-- 1ª demostración
example
  (h1 : ∀x, P x → ¬Q x)
  (h2 : ∀x, P x)
  : ∀x, ¬Q x :=
assume x₀,
have h4 : P x₀ → ¬Q x₀, from h1 x₀,
have h5 : P x₀,         from h2 x₀,
show ¬Q x₀,             from h4 h5

-- 2ª demostración
example
  (h1 : ∀x, P x → ¬Q x)
  (h2 : ∀x, P x)
  : ∀x, ¬Q x :=
assume x₀, (h1 x₀) (h2 x₀)

-- 3ª demostración
example
  (h1 : ∀x, P x → ¬Q x)
  (h2 : ∀x, P x)
  : ∀x, ¬Q x :=
λ x₀, (h1 x₀) (h2 x₀)

-- 4ª demostración
example
  (h1 : ∀x, P x → ¬Q x)
  (h2 : ∀x, P x)
  : ∀x, ¬Q x :=
begin
```

```
  intro x₀,
  apply h1,
  apply h2,
end

-- 5ª demostración
example
  (h1 : ∀x, P x → ¬Q x)
  (h2 : ∀x, P x)
  : ∀x, ¬Q x :=
begin
  intro x₀,
  specialize h1 x₀,
  specialize h2 x₀,
  apply h1,
  exact h2,
end

-- 6ª demostración
example
  (h1 : ∀x, P x → ¬Q x)
  (h2 : ∀x, P x)
  : ∀x, ¬Q x :=
begin
  intro x₀,
  specialize h1 x₀,
  specialize h2 x₀,
  exact h1 h2,
end

-- 7ª demostración
example
  (h1 : ∀x, P x → ¬Q x)
  (h2 : ∀x, P x)
  : ∀x, ¬Q x :=
-- by hint
by tauto

-- 8ª demostración
example
  (h1 : ∀x, P x → ¬Q x)
  (h2 : ∀x, P x)
  : ∀x, ¬Q x :=
by finish
```

3.2. Reglas del cuantificador existencial

3.2.1. Regla de introducción del cuantificador existencial

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de introducción del cuantificador existencial
-- ====================================================

-- Ej. 1. Demostrar
--    ∀x P(x) ⊢ ∃x P(x)

import tactic

variable U : Type
variable c : U
variable P : U -> Prop

-- 1ª demostración
example
  (h1 : ∀x, P x)
  : ∃x, P x :=
have h2 : P c, from h1 c,
show ∃x, P x,  from exists.intro c h2

-- 2ª demostración
example
  (h1 : ∀x, P x)
  : ∃x, P x :=
show ∃x, P x,  from exists.intro c (h1 c)

-- 3ª demostración
example
  (h1 : ∀x, P x)
  : ∃x, P x :=
exists.intro c (h1 c)

-- 4ª demostración
example
  (h1 : ∀x, P x)
  : ∃x, P x :=
⟨c, h1 c⟩
```

```
-- 5ª demostración
example
  (a : U)
  (h1 : ∀x, P x)
  : ∃x, P x :=
begin
  use a,
  apply h1,
end

-- 6ª demostración
example
  (a : U)
  (h1 : ∀x, P x)
  : ∃x, P x :=
begin
  constructor,
  apply h1 a,
end

-- 7ª demostración
example
  [inhabited U]
  (h1 : ∀x, P x)
  : ∃x, P x :=
begin
  constructor,
  apply h1 (default U),
end

-- 8ª demostración
example
  (h : nonempty U)
  (h1 : ∀x, P x)
  : ∃x, P x :=
begin
  use (classical.choice h),
  apply h1,
end
```

3.2.2. Regla de eliminación del cuantificador existencial

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de eliminación del cuantificador existencial
-- ===================================================

-- Ej. 1. Demostrar
--    ∀x [P(x) → Q(x)], ∃x P(x) ⊢ ∃x Q(x)

import tactic

variable  U : Type
variables P Q : U -> Prop

-- 1ª demostración
-- ===============

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
exists.elim h2
  ( assume x₀ (h3 : P x₀),
    have h4 : P x₀ → Q x₀, from h1 x₀,
    have h5 : Q x₀,        from h4 h3,
    show ∃x, Q x,          from exists.intro x₀ h5 )

-- 2ª demostración
-- ===============

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
exists.elim h2
  ( assume x₀ (h3 : P x₀),
    have h4 : P x₀ → Q x₀, from h1 x₀,
    have h5 : Q x₀,        from h4 h3,
    show ∃x, Q x,          from ⟨x₀, h5⟩ )

-- 3ª demostración
-- ===============

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
exists.elim h2
```

```
  ( assume x₀ (h3 : P x₀),
    have h4 : P x₀ → Q x₀, from h1 x₀,
    have h5 : Q x₀,        from h4 h3,
    ⟨x₀, h5⟩ )

-- 4ª demostración
-- ===============

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
exists.elim h2
  ( assume x₀ (h3 : P x₀),
    have h4 : P x₀ → Q x₀, from h1 x₀,
    ⟨x₀, h4 h3⟩ )

-- 5ª demostración
-- ===============

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
exists.elim h2
  ( assume x₀ (h3 : P x₀),
    ⟨x₀, h1 x₀ h3⟩ )

-- 6ª demostración
-- ===============

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
exists.elim h2 (λ x₀ h3, ⟨x₀, h1 x₀ h3⟩)

-- 7ª demostración
-- ===============

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
-- by library_search
```

```
Exists.imp h1 h2

-- 8ª demostración
-- ===============

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
begin
  cases h2 with x₀ h3,
  use x₀,
  apply h1,
  exact h3,
end

-- 9ª demostración
-- ===============

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
begin
  cases h2 with x₀ h3,
  use x₀,
  specialize h1 x₀,
  apply h1,
  exact h3,
end

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

example
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
-- by hint
by tauto

-- 11ª demostración
-- ===============

example
```

```
  (h1 : ∀x, P x → Q x)
  (h2 : ∃x, P x)
  : ∃x, Q x :=
by finish
```

3.3. Ejercicios sobre cuantificadores

3.3.1. Pruebas de ¬∀x P(x) ↔ ∃x ¬P(x)

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de ¬∀x P(x) ↔ ∃x ¬P(x)
-- ==============================

import tactic

variable {U : Type}
variable {P : U -> Prop}

-- ----------------------------------------------------------
-- Ej. 1. Demostrar que
--    ¬∀x P(x) ⊢ ∃x ¬P(x)
-- ----------------------------------------------------------

-- 1ª demostración
example
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
by_contra
  ( assume h2 : ¬∃x, ¬P x,
    have h8 : ∀x, P x, from
      ( assume x₀,
        show P x₀, from
          by_contra
            ( assume h4 : ¬P x₀,
              have h5 : ∃x, ¬P x, from exists.intro x₀ h4,
              show false, from h2 h5 )),
    show false, from h1 h8)

-- 2ª demostración
example
  (h1 : ¬∀x, P x)
```

```
  : ∃x, ¬P x :=
by_contra
  ( assume h2 : ¬∃x, ¬P x,
    have h8 : ∀x, P x, from
      ( assume x₀,
        show P x₀, from
          by_contra
            ( assume h4 : ¬P x₀,
              have h5 : ∃x, ¬P x, from exists.intro x₀ h4,
              show false, from h2 h5 )),
    h1 h8)

-- 3ª demostración
example
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
by_contra
  ( assume h2 : ¬∃x, ¬P x,
    have h8 : ∀x, P x, from
      ( assume x₀,
        show P x₀, from
          by_contra
            ( assume h4 : ¬P x₀,
              have h5 : ∃x, ¬P x, from exists.intro x₀ h4,
              h2 h5 )),
    h1 h8)

-- 4ª demostración
example
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
by_contra
  ( assume h2 : ¬∃x, ¬P x,
    have h8 : ∀x, P x, from
      ( assume x₀,
        show P x₀, from
          by_contra
            ( assume h4 : ¬P x₀,
              have h5 : ∃x, ¬P x, from ⟨x₀, h4⟩,
              h2 h5 )),
    h1 h8)

-- 5ª demostración
example
  (h1 : ¬∀x, P x)
```

```
  : ∃x, ¬P x :=
by_contra
  ( assume h2 : ¬∃x, ¬P x,
    have h8 : ∀x, P x, from
      ( assume x₀,
        show P x₀, from
          by_contra
            ( assume h4 : ¬P x₀,
              h2 ⟨x₀, h4⟩ )),
    h1 h8)

-- 6ª demostración
example
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
by_contra
  ( assume h2 : ¬∃x, ¬P x,
    have h8 : ∀x, P x, from
      ( assume x₀,
        show P x₀, from
          by_contra (λ h4, h2 ⟨x₀, h4⟩)),
    h1 h8)

-- 7ª demostración
example
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
by_contra
  ( assume h2 : ¬∃x, ¬P x,
    have h8 : ∀x, P x, from
      ( assume x₀,
        by_contra (λ h4, h2 ⟨x₀, h4⟩)),
    h1 h8)

-- 8ª demostración
example
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
by_contra
  ( assume h2 : ¬∃x, ¬P x,
    have h8 : ∀x, P x, from
      (λ x₀, by_contra (λ h4, h2 ⟨x₀, h4⟩)),
    h1 h8)

-- 9ª demostración
```

```
example
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
by_contra
  ( assume h2 : ¬∃x, ¬P x,
    h1 (λ x₀, by_contra (λ h4, h2 ⟨x₀, h4⟩)))

-- 10ª demostración
example
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
by_contra (λ h2, h1 (λ x₀, by_contra (λ h4, h2 ⟨x₀, h4⟩)))

-- 11ª demostración
example
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
-- by library_search
not_forall.mp h1

-- 12ª demostración
lemma aux1
  (h1 : ¬∀x, P x)
  : ∃x, ¬P x :=
-- by hint
by finish

-- ----------------------------------------------------
-- Ej. 2. Demostrar que
--    ∃x ¬P(x) ⊢ ¬∀x P(x)
-- ----------------------------------------------------

-- 1ª demostración
example
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
assume h2 : ∀x, P x,
exists.elim h1
  ( assume x₀ (h3 : ¬P x₀),
    have h4 : P x₀, from h2 x₀,
    show false, from h3 h4)

-- 2ª demostración
example
  (h1 : ∃x, ¬P x)
```

```
  : ¬∀x, P x :=
assume h2 : ∀x, P x,
exists.elim h1
  ( assume x₀ (h3 : ¬P x₀),
    have h4 : P x₀, from h2 x₀,
    h3 h4)

-- 3ª demostración
example
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
assume h2 : ∀x, P x,
exists.elim h1
  ( assume x₀ (h3 : ¬P x₀),
    h3 (h2 x₀))

-- 4ª demostración
example
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
assume h2 : ∀x, P x,
exists.elim h1
  (λ x₀ h3, h3 (h2 x₀))

-- 5ª demostración
example
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
λ h2, exists.elim h1 (λ x₀ h3, h3 (h2 x₀))

-- 6ª demostración
example
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
-- by library_search
not_forall.mpr h1

-- 7ª demostración
example
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
assume h2 : ∀x, P x,
match h1 with ⟨x₀, (h3 : ¬P x₀)⟩ :=
  ( have h4 : P x₀, from h2 x₀,
    show false,     from h3 h4)
```

```
end

-- 8ª demostración
example
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
begin
  intro h2,
  cases h1 with x₀ h3,
  apply h3,
  apply h2,
end

example
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
begin
  intro h2,
  obtain ⟨x₀, h3⟩ := h1,
  apply h3,
  apply h2,
end

-- 9ª demostración
example
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
-- by hint
by tauto

-- 10ª demostración
lemma aux2
  (h1 : ∃x, ¬P x)
  : ¬∀x, P x :=
by finish

#print axioms aux2

-- ----------------------------------------------------
-- Ej. 3. Demostrar que
--    ¬∀x P(x) ↔ ∃x ¬P(x)
-- ----------------------------------------------------

-- 1ª demostración
example :
```

```
  (¬∀x, P x) ↔ (∃x, ¬P x) :=
iff.intro
  ( assume h1 : ¬∀x, P x,
    show ∃x, ¬P x, from aux1 h1)
  ( assume h2 : ∃x, ¬P x,
    show ¬∀x, P x, from aux2 h2)

-- 2ª demostración
example :
  (¬∀x, P x) ↔ (∃x, ¬P x) :=
iff.intro aux1 aux2

-- 3ª demostración
example :
  (¬∀x, P x) ↔ (∃x, ¬P x) :=
-- by library_search
not_forall

-- 4ª demostración
example :
  (¬∀x, P x) ↔ (∃x, ¬P x) :=
begin
  split,
  { exact aux1, },
  { exact aux2, },
end

-- 5ª demostración
example :
  (¬∀x, P x) ↔ (∃x, ¬P x) :=
-- by hint
by finish
```

3.3.2. Pruebas de ∀x (P(x) ∧ Q(x)) ↔ ∀x P(x) ∧ ∀x Q(x)

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de ∀x (P(x) ∧ Q(x)) ↔ ∀x P(x) ∧ ∀x Q(x)
-- ================================================

import tactic

section
```

```
variable  {U : Type}
variables {P Q : U -> Prop}

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    ∀x (P(x) ∧ Q(x)) ⊢ ∀x P(x) ∧ ∀x Q(x)
-- ----------------------------------------------------

-- 1ª demostración
example
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
have h5 : ∀x, P x, from
    assume x₀,
    have h3 : P x₀ ∧ Q x₀,  from h1 x₀,
    show P x₀,              from and.elim_left h3,
have h9 : ∀x, Q x, from
    assume x₁,
    have h7 : P x₁ ∧ Q x₁,  from h1 x₁,
    show Q x₁,              from and.elim_right h7,
show (∀x, P x) ∧ (∀x, Q x), from and.intro h5 h9

-- 2ª demostración
example
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
have h5 : ∀x, P x, from
    assume x₀,
    have h3 : P x₀ ∧ Q x₀,  from h1 x₀,
    show P x₀,              from h3.left,
have h9 : ∀x, Q x, from
    assume x₁,
    have h7 : P x₁ ∧ Q x₁,  from h1 x₁,
    show Q x₁,              from h7.right,
show (∀x, P x) ∧ (∀x, Q x), from ⟨h5, h9⟩

-- 3ª demostración
example
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
have h5 : ∀x, P x, from
    assume x₀,
    have h3 : P x₀ ∧ Q x₀,  from h1 x₀,
    h3.left,
```

```
have h9 : ∀x, Q x, from
    assume x₁,
    have h7 : P x₁ ∧ Q x₁,  from h1 x₁,
    h7.right,
show (∀x, P x) ∧ (∀x, Q x), from ⟨h5, h9⟩

-- 4ª demostración
example
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
have h5 : ∀x, P x, from
    assume x₀,
    (h1 x₀).left,
have h9 : ∀x, Q x, from
    assume x₁,
    (h1 x₁).right,
show (∀x, P x) ∧ (∀x, Q x), from ⟨h5, h9⟩

-- 5ª demostración
example
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
have h5 : ∀x, P x, from
    λ x₀, (h1 x₀).left,
have h9 : ∀x, Q x, from
    λ x₁, (h1 x₁).right,
show (∀x, P x) ∧ (∀x, Q x), from ⟨h5, h9⟩

-- 6ª demostración
example
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
have h5 : ∀x, P x, from
    λ x₀, (h1 x₀).left,
have h9 : ∀x, Q x, from
    λ x₁, (h1 x₁).right,
⟨h5, h9⟩

-- 7ª demostración
example
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
⟨λ x₀, (h1 x₀).left, λ x₁, (h1 x₁).right⟩

-- 8ª demostración
```

```
example
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
-- by library_search
forall_and_distrib.mp h1

-- 9ª demostración
example
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
begin
  split,
  { intro x₀,
    specialize h1 x₀,
    exact h1.left, },
  { intro x₁,
    specialize h1 x₁,
    exact h1.right, },
end

-- 9ª demostración
lemma aux1
  (h1 : ∀x, P x ∧ Q x)
  : (∀x, P x) ∧ (∀x, Q x) :=
-- by hint
by finish

-- ------------------------------------------------------
-- Ej. 2. Demostrar
--    ∀x P(x) ∧ ∀x Q(x) ⊢ ∀x (P(x) ∧ Q(x))
-- ------------------------------------------------------

-- 1ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
assume x₀,
have h3 : ∀x, P x, from and.elim_left h1,
have h4 : P x₀,    from h3 x₀,
have h5 : ∀x, Q x, from and.elim_right h1,
have h6 : Q x₀,    from h5 x₀,
show P x₀ ∧ Q x₀,  from and.intro h4 h6

-- 2ª demostración
example
```

```
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
assume x₀,
have h3 : ∀x, P x, from h1.left,
have h4 : P x₀,    from h3 x₀,
have h5 : ∀x, Q x, from h1.right,
have h6 : Q x₀,    from h5 x₀,
show P x₀ ∧ Q x₀,  from ⟨h4, h6⟩

-- 3ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
assume x₀,
have h3 : ∀x, P x, from h1.left,
have h4 : P x₀,    from h3 x₀,
have h5 : ∀x, Q x, from h1.right,
have h6 : Q x₀,    from h5 x₀,
⟨h4, h6⟩

-- 4ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
assume x₀,
have h3 : ∀x, P x, from h1.left,
have h4 : P x₀,    from h3 x₀,
have h5 : ∀x, Q x, from h1.right,
⟨h4, h5 x₀⟩

-- 5ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
assume x₀,
have h3 : ∀x, P x, from h1.left,
have h4 : P x₀,    from h3 x₀,
⟨h4, h1.right x₀⟩

-- 6ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
assume x₀,
have h3 : ∀x, P x, from h1.left,
```

```
⟨h3 x₀, h1.right x₀⟩

-- 7ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
assume x₀,
⟨h1.left x₀, h1.right x₀⟩

-- 8ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
λ x₀, ⟨h1.left x₀, h1.right x₀⟩

-- 9ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
-- by library_search
forall_and_distrib.mpr h1

-- 10ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
begin
  cases h1 with h2 h3,
  intro x₀,
  split,
  { apply h2, },
  { apply h3, },
end

-- 11ª demostración
example
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
-- by hint
by tauto

-- 12ª demostración
lemma aux2
  (h1 : (∀x, P x) ∧ (∀x, Q x))
  : ∀x, P x ∧ Q x :=
```

```
by finish

-- ----------------------------------------------------
-- Ej. 3. Demostrar
--    ∀x (P(x) ∧ Q(x)) ↔ ∀x P(x) ∧ ∀x Q(x)
-- ----------------------------------------------------

-- 1ª demostración
example :
  (∀x, P x ∧ Q x) ↔ (∀x, P x) ∧ (∀x, Q x) :=
iff.intro aux1 aux2

-- 2ª demostración
example :
  (∀x, P x ∧ Q x) ↔ (∀x, P x) ∧ (∀x, Q x) :=
-- by library_search
forall_and_distrib

-- 3ª demostración
example :
  (∀x, P x ∧ Q x) ↔ (∀x, P x) ∧ (∀x, Q x) :=
-- by hint
by finish

end
```

3.3.3. Pruebas de ∃x (P(x) ∨ Q(x)) ↔ ∃x P(x) ∨ ∃x Q(x)

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de ∃x (P(x) ∨ Q(x)) ↔ ∃x P(x) ∨ ∃x Q(x)
-- ===============================================

import tactic

section

variable  {U : Type}
variables {P Q : U -> Prop}

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    ∃x (P(x) ∨ Q(x)) ⊢ ∃x P(x) ∨ ∃x Q(x)
```

```
-- ----------------------------------------------------

-- 1ª demostración
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
exists.elim h1
  ( assume x₀ (h2 : P x₀ ∨ Q x₀),
    or.elim h2
    ( assume h3 : P x₀,
      have h4 : ∃x, P x,           from exists.intro x₀ h3,
      show (∃x, P x) ∨ (∃x, Q x), from or.inl h4 )
    ( assume h6 : Q x₀,
      have h7 : ∃x, Q x,           from exists.intro x₀ h6,
      show (∃x, P x) ∨ (∃x, Q x), from or.inr h7 ))

-- 2ª demostración
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
exists.elim h1
  ( assume x₀ (h2 : P x₀ ∨ Q x₀),
    or.elim h2
    ( assume h3 : P x₀,
      have h4 : ∃x, P x,           from ⟨x₀, h3⟩,
      show (∃x, P x) ∨ (∃x, Q x), from or.inl h4 )
    ( assume h6 : Q x₀,
      have h7 : ∃x, Q x,           from ⟨x₀, h6⟩,
      show (∃x, P x) ∨ (∃x, Q x), from or.inr h7 ))

-- 3ª demostración
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
exists.elim h1
  ( assume x₀ (h2 : P x₀ ∨ Q x₀),
    or.elim h2
    ( assume h3 : P x₀,
      have h4 : ∃x, P x,           from ⟨x₀, h3⟩,
      or.inl h4 )
    ( assume h6 : Q x₀,
      have h7 : ∃x, Q x,           from ⟨x₀, h6⟩,
      or.inr h7 ))

-- 4ª demostración
```

```
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
exists.elim h1
  ( assume x₀ (h2 : P x₀ ∨ Q x₀),
    or.elim h2
    ( assume h3 : P x₀,
      or.inl ⟨x₀, h3⟩ )
    ( assume h6 : Q x₀,
      or.inr ⟨x₀, h6⟩ ))

-- 5ª demostración
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
exists.elim h1
  ( assume x₀ (h2 : P x₀ ∨ Q x₀),
    or.elim h2
    ( λ h3, or.inl ⟨x₀, h3⟩ )
    ( λ h6, or.inr ⟨x₀, h6⟩ ))

-- 6ª demostración
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
exists.elim h1
  (λ x₀ h2, h2.elim (λ h3, or.inl ⟨x₀, h3⟩)
                    (λ h6, or.inr ⟨x₀, h6⟩))

-- 7ª demostración
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
-- by library_search
exists_or_distrib.mp h1

-- 8ª demostración
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
match h1 with ⟨x₀, (h2 : P x₀ ∨ Q x₀)⟩ :=
  ( or.elim h2
    ( assume h3 : P x₀,
      have h4 : ∃x, P x,           from exists.intro x₀ h3,
      show (∃x, P x) ∨ (∃x, Q x), from or.inl h4 )
```

```
    ( assume h6 : Q x₀,
      have h7 : ∃x, Q x,            from exists.intro x₀ h6,
      show (∃x, P x) ∨ (∃x, Q x), from or.inr h7 ))
end

-- 9ª demostración
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
begin
  cases h1 with x₀ h3,
  cases h3 with hp hq,
  { left,
    use x₀,
    exact hp, },
  { right,
    use x₀,
    exact hq, },
end

-- 10ª demostración
example
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
begin
  rcases h1 with ⟨x₀, hp | hq⟩,
  { left,
    use x₀,
    exact hp, },
  { right,
    use x₀,
    exact hq, },
end

-- 11ª demostración
lemma aux1
  (h1 : ∃x, P x  ∨ Q x)
  : (∃x, P x) ∨ (∃x, Q x) :=
-- by hint
by finish

-- ----------------------------------------------------
-- Ej. 2. Demostrar
--     ∃x P(x) ∨ ∃x Q(x) ⊢ ∃x (P(x) ∨ Q(x))
-- ----------------------------------------------------
```

```
-- 1ª demostración
example
  (h1 : (∃x, P x) ∨ (∃x, Q x))
  : ∃x, P x  ∨ Q x :=
or.elim h1
  ( assume h2 : ∃x, P x,
    exists.elim h2
      ( assume x₀ (h3 : P x₀),
        have h4 : P x₀ ∨ Q x₀, from or.inl h3,
        show ∃x, P x  ∨ Q x,   from exists.intro x₀ h4 ))
  ( assume h2 : ∃x, Q x,
    exists.elim h2
      ( assume x₀ (h3 : Q x₀),
        have h4 : P x₀ ∨ Q x₀, from or.inr h3,
        show ∃x, P x  ∨ Q x,   from exists.intro x₀ h4 ))

-- 2ª demostración
example
  (h1 : (∃x, P x) ∨ (∃x, Q x))
  : ∃x, P x  ∨ Q x :=
h1.elim
  ( assume ⟨x₀, (h3 : P x₀)⟩,
    have h4 : P x₀ ∨ Q x₀, from or.inl h3,
    show ∃x, P x  ∨ Q x,   from ⟨x₀, h4⟩ )
  ( assume ⟨x₀, (h3 : Q x₀)⟩,
    have h4 : P x₀ ∨ Q x₀, from or.inr h3,
    show ∃x, P x  ∨ Q x,   from ⟨x₀, h4⟩ )

-- 3ª demostración
example
  (h1 : (∃x, P x) ∨ (∃x, Q x))
  : ∃x, P x  ∨ Q x :=
h1.elim
  ( assume ⟨x₀, (h3 : P x₀)⟩,
    have h4 : P x₀ ∨ Q x₀, from or.inl h3,
    ⟨x₀, h4⟩ )
  ( assume ⟨x₀, (h3 : Q x₀)⟩,
    have h4 : P x₀ ∨ Q x₀, from or.inr h3,
    ⟨x₀, h4⟩ )

-- 4ª demostración
example
  (h1 : (∃x, P x) ∨ (∃x, Q x))
  : ∃x, P x  ∨ Q x :=
```

```
h1.elim
  ( assume ⟨x₀, (h3 : P x₀)⟩,
    ⟨x₀, or.inl h3⟩ )
  ( assume ⟨x₀, (h3 : Q x₀)⟩,
    ⟨x₀, or.inr h3⟩ )

-- 5ª demostración
example
  (h1 : (∃x, P x) ∨ (∃x, Q x))
  : ∃x, P x  ∨ Q x :=
h1.elim
  (λ ⟨x₀, h3⟩, ⟨x₀, or.inl h3⟩)
  (λ ⟨x₀, h3⟩, ⟨x₀, or.inr h3⟩)

-- 6ª demostración
example
  (h1 : (∃x, P x) ∨ (∃x, Q x))
  : ∃x, P x  ∨ Q x :=
-- by library_search
exists_or_distrib.mpr h1

-- 7ª demostración
example
  (h1 : (∃x, P x) ∨ (∃x, Q x))
  : ∃x, P x  ∨ Q x :=
begin
  cases h1 with hp hq,
  { cases hp with x₀ hx₀,
    use x₀,
    left,
    exact hx₀, },
  { cases hq with x₁ hx₁,
    use x₁,
    right,
    exact hx₁, },
end

-- 8ª demostración
example
  (h1 : (∃x, P x) ∨ (∃x, Q x))
  : ∃x, P x  ∨ Q x :=
begin
  rcases h1 with ⟨x₀, hx₀⟩ | ⟨x₁, hx₁⟩,
  { use x₀,
    left,
```

```
    exact hx₀, },
  { use x₁,
    right,
    exact hx₁, },
end

-- 9ª demostración
lemma aux2
  (h1 : (∃x, P x) ∨ (∃x, Q x))
  : ∃x, P x  ∨ Q x :=
-- by hint
by finish

-- ----------------------------------------------------
-- Ej. 3. Demostrar
--    ∃x (P(x) ∨ Q(x)) ↔ ∃x P(x) ∨ ∃x Q(x)
-- ----------------------------------------------------

-- 1ª demostración
example :
  (∃x, P x  ∨ Q x) ↔ (∃x, P x) ∨ (∃x, Q x) :=
iff.intro aux1 aux2

-- 2ª demostración
example :
  (∃x, P x  ∨ Q x) ↔ (∃x, P x) ∨ (∃x, Q x) :=
⟨aux1, aux2⟩

-- 3ª demostración
example :
  (∃x, P x  ∨ Q x) ↔ (∃x, P x) ∨ (∃x, Q x) :=
-- by library_search
exists_or_distrib

end
```

3.3.4. Pruebas de ∃x∃y P(x,y) ↔ ∃y∃x P(x,y)

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de ∃x∃yP(x,y) ↔ ∃y∃xP(x,y)
-- ==================================
```

```
import tactic

section

variable {U : Type}
variable {P : U -> U -> Prop}

-- ------------------------------------------------------
-- Ej. 1. Demostrar que
--    ∃x∃yP(x,y) → ∃y∃xP(x,y)
-- ------------------------------------------------------

-- 1ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
assume h1 : ∃ x, ∃ y, P x y,
exists.elim h1
  ( assume x₀ (h2 : ∃ y, P x₀ y),
    exists.elim h2
      ( assume y₀ (h3 : P x₀ y₀),
        have h4 : ∃ x, P x y₀, from exists.intro x₀ h3,
        show ∃ y, ∃ x, P x y,  from exists.intro y₀ h4))

-- 2ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
assume ⟨x₀, y₀, (h1 : P x₀ y₀)⟩,
have h2 : ∃ x, P x y₀, from ⟨x₀, h1⟩,
show (∃ y, ∃ x, P x y), from ⟨y₀, h2⟩

-- 3ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
assume ⟨x₀, y₀, (h1 : P x₀ y₀)⟩,
show (∃ y, ∃ x, P x y), from ⟨y₀, ⟨x₀, h1⟩⟩

-- 4ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
assume ⟨x₀, y₀, (h1 : P x₀ y₀)⟩,
show (∃ y, ∃ x, P x y), from ⟨y₀, x₀, h1⟩

-- 5ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
```

```
assume ⟨x₀, y₀, (h1 : P x₀ y₀)⟩,
⟨y₀, x₀, h1⟩

-- 6ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
λ ⟨x₀, y₀, h1⟩, ⟨y₀, x₀, h1⟩

-- 7ª demostración
example :
  (∃ x y, P x y) → (∃ y x, P x y) :=
-- by library_search
exists_comm.mp

-- 8ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
begin
  intro h1,
  cases h1 with x₀ h2,
  cases h2 with y₀ h3,
  use y₀,
  use x₀,
  exact h3,
end

-- 9ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
begin
  intro h1,
  cases h1 with x₀ h2,
  cases h2 with y₀ h3,
  use [y₀, x₀],
  exact h3,
end

-- 10ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
begin
  intro h1,
  rcases h1 with ⟨x₀, y₀, h2⟩,
  use [y₀, x₀],
  exact h2,
```

```
end

-- 11ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
begin
  intro h1,
  rcases h1 with ⟨x₀, y₀, h2⟩,
  exact ⟨y₀, x₀, h2⟩,
end

-- 12ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
begin
  rintro ⟨x₀, y₀, h2⟩,
  exact ⟨y₀, x₀, h2⟩,
end

-- 13ª demostración
example :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
-- by hint
by tauto

-- 14ª demostración
lemma aux :
  (∃ x, ∃ y, P x y) → (∃ y, ∃ x, P x y) :=
by finish

-- ----------------------------------------------------
-- Ej. 2. Demostrar que
--    ∃x∃yP(x,y) ↔ ∃y∃xP(x,y)
-- ----------------------------------------------------

-- 1ª demostración
example :
  (∃ x y, P x y) ↔ (∃ y x, P x y) :=
iff.intro aux aux

-- 2ª demostración
example :
  (∃ x y, P x y) ↔ (∃ y x, P x y) :=
⟨aux, aux⟩
```

```
-- 3ª demostración
example :
  (∃ x y, P x y) ↔ (∃ y x, P x y) :=
-- by library_search
exists_comm

-- 4ª demostración
example :
  (∃ x y, P x y) ↔ (∃ y x, P x y) :=
-- by hint
by tauto

end
```

3.4. Reglas de la igualdad

3.4.1. Regla de eliminación de la igualdad

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de eliminación de la igualdad
-- ===================================

import tactic

#check @eq.subst

-- ----------------------------------------------------
-- Ej. 1. Demostrar que
--      x + 1 = 1 + x
--      x + 1 > 1 → x + 1 > 0
--      ⊢ 1 + x > 1 → 1 + x > 0
-- ----------------------------------------------------

variable (U : Type)
variable (x : U)
variable [has_add U]
variable [has_one U]
variable [has_lt U]
variable [has_zero U]

-- 1ª demostración
```

```
example
  (h1 : x + 1 = 1 + x)
  (h2 : x + 1 > 1 → x + 1 > 0)
  : 1 + x > 1 → 1 + x > 0 :=
eq.subst h1 h2


-- 2ª demostración
example
  (h1 : x + 1 = 1 + x)
  (h2 : x + 1 > 1 → x + 1 > 0)
  : 1 + x > 1 → 1 + x > 0 :=
h1 ▸ h2

-- 3ª demostración
example
  (h1 : x + 1 = 1 + x)
  (h2 : x + 1 > 1 → x + 1 > 0)
  : 1 + x > 1 → 1 + x > 0 :=
begin
  rw h1 at h2,
  exact h2,
end

-- 4ª demostración
example
  (h1 : x + 1 = 1 + x)
  (h2 : x + 1 > 1 → x + 1 > 0)
  : 1 + x > 1 → 1 + x > 0 :=
begin
  rw ←h1,
  exact h2,
end

-- 5ª demostración
example
  (h1 : x + 1 = 1 + x)
  (h2 : x + 1 > 1 → x + 1 > 0)
  : 1 + x > 1 → 1 + x > 0 :=
-- by hint
by finish
```

3.4.2. Pruebas de la transitividad de la igualdad

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de la transitividad de la igualdad
-- ==========================================

import tactic

variable  (U : Type)
variables (x y z : U)

-- ----------------------------------------------------
-- Ej. 1. Demostrar que
--    x = y, y = z ⊢ x = z
-- ----------------------------------------------------

-- 1ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
eq.subst h2 h1

-- 2ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
h2 ▸ h1

-- 3ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
eq.substr h1 h2

-- 4ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
eq.trans h1 h2
```

```
-- 5ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
begin
  rw h1,
  exact h2,
end

-- 6ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
begin
  rw h1,
  assumption,
end

-- 7ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
begin
  rwa h1,
end

-- 8ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
by rwa h1

-- 9ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
begin
  rwa ←h2,
end
```

```
-- 10ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
begin
  rwa h2 at h1,
end

-- 11ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
by simp *

-- 12ª demostración
example
  (h1 : x = y)
  (h2 : y = z)
  : x = z :=
-- by hint
by finish
```

3.4.3. Regla de introducción de la igualdad

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de introducción de la igualdad
-- ====================================

import tactic

#check @eq.refl

variable  (U : Type)
variables (x y : U)

-- ------------------------------------------------------
-- Ej. Demostrar
--    x = y ⊢ y = x
-- ------------------------------------------------------
```

```
-- 1ª demostración
example
  (h1 : x = y)
  : y = x :=
have h2 : x = x, from eq.refl x,
show y = x,      from eq.subst h1 h2

-- 2ª demostración
example
  (h1 : x = y)
  : y = x :=
have h2 : x = x, from eq.refl x,
show y = x,      from h1 ▸ h2

-- 3ª demostración
example
  (h1 : x = y)
  : y = x :=
have h2 : x = x, from eq.refl x,
h1 ▸ h2

-- 4ª demostración
example
  (h1 : x = y)
  : y = x :=
h1 ▸ eq.refl x

-- 5ª demostración
example
  (h1 : x = y)
  : y = x :=
h1 ▸ rfl

-- 6ª demostración
example
  (h1 : x = y)
  : y = x :=
-- by library_search
eq.symm h1

-- 7ª demostración
example
  (h1 : x = y)
  : y = x :=
begin
```

```
  rw h1,
end

-- 8ª demostración
example
  (h1 : x = y)
  : y = x :=
by rw h1

-- 9ª demostración
example
  (h1 : x = y)
  : y = x :=
by simp *

-- 10ª demostración
example
  (h1 : x = y)
  : y = x :=
-- by hint
by tauto

-- 11ª demostración
example
  (h1 : x = y)
  : y = x :=
by finish

-- 12ª demostración
example
  (h1 : x = y)
  : y = x :=
by solve_by_elim
```

3.4.4. Pruebas de y = x → y = z → x = z

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de y = x → y = z → x = z
-- ================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
```

```
--    y = x → y = z → x = z
-- ----------------------------------------------------

import tactic

variables (U : Type)
variables (x y z : U)

-- 1ª demostración
example : y = x → y = z → x = z :=
assume h1 : y = x,
assume h2 : y = z,
have h3 : x = y, from eq.symm h1,
show x = z,      from eq.trans h3 h2

-- 2ª demostración
example : y = x → y = z → x = z :=
assume h1 : y = x,
assume h2 : y = z,
have h3 : x = y, from eq.symm h1,
eq.trans h3 h2

-- 3ª demostración
example : y = x → y = z → x = z :=
assume h1 : y = x,
assume h2 : y = z,
eq.trans (eq.symm h1) h2

-- 4ª demostración
example : y = x → y = z → x = z :=
λ h1 h2, eq.trans (eq.symm h1) h2

-- 5ª demostración
example : y = x → y = z → x = z :=
λ h1 h2, eq.trans h1.symm h2

-- 6ª demostración
example : y = x → y = z → x = z :=
-- by library_search
λ h, h.congr_left.mp

-- 7ª demostración
example : y = x → y = z → x = z :=
begin
  intros h1 h2,
```

```
  rwa ←h1,
end

-- 8ª demostración
example : y = x → y = z → x = z :=
begin
  intros h1 h2,
  rw h1 at h2,
  assumption,
end

-- 9ª demostración
example : y = x → y = z → x = z :=
begin
  intros h1 h2,
  rwa h1 at h2,
end

-- 10ª demostración
example : y = x → y = z → x = z :=
begin
  intros h1 h2,
  calc x = y : h1.symm
     ... = z : h2,
end

-- 11ª demostración
example : y = x → y = z → x = z :=
-- by hint
by finish

-- 12ª demostración
example : y = x → y = z → x = z :=
assume h1 : y = x,
assume h2 : y = z,
show x = z,
  begin
    rw ←h1,
    rw h2
  end

-- 13ª demostración
example : y = x → y = z → x = z :=
assume h1 : y = x,
assume h2 : y = z,
```

```
show x = z,
  begin
    rw [←h1, h2]
  end

-- 14ª demostración
example : y = x → y = z → x = z :=
assume h1 : y = x,
assume h2 : y = z,
show x = z, by rw [←h1, h2]
```

3.4.5. Pruebas de (x + y) + z = (x + z) + y

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de (x + y) + z = (x + z) + y
-- ====================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar que para todo x, y, z en ℤ se tiene
--    (x + y) + z = (x + z) + y
-- ----------------------------------------------------

import tactic
import data.int.basic

variables (x y z : ℤ)

-- 1ª demostración
example : (x + y) + z = (x + z) + y :=
calc
   (x + y) + z = x + (y + z) : add_assoc x y z
           ... = x + (z + y) : eq.subst (add_comm y z) rfl
           ... = (x + z) + y : eq.symm (add_assoc x z y)

-- 2ª demostración
example : (x + y) + z = (x + z) + y :=
calc
  (x + y) + z = x + (y + z) : by rw add_assoc
          ... = x + (z + y) : by rw [add_comm y z]
          ... = (x + z) + y : by rw add_assoc
```

```
-- 3ª demostración
example : (x + y) + z = (x + z) + y :=
begin
  rw add_assoc,
  rw add_comm y z,
  rw add_assoc,
end

example : (x + y) + z = (x + z) + y :=
begin
  rw [add_assoc, add_comm y z, add_assoc],
end

-- 4ª demostración
example : (x + y) + z = (x + z) + y :=
by rw [add_assoc, add_comm y z, add_assoc]

-- 5ª demostración
example : (x + y) + z = (x + z) + y :=
-- by library_search
add_right_comm x y z

-- 6ª demostración
example : (x + y) + z = (x + z) + y :=
-- by hint
by omega

-- 7ª demostración
example : (x + y) + z = (x + z) + y :=
by linarith

-- 8ª demostración
example : (x + y) + z = (x + z) + y :=
by nlinarith

-- 9ª demostración
example : (x + y) + z = (x + z) + y :=
by ring

-- 10ª demostración
example : (x + y) + z = (x + z) + y :=
by finish
```

3.4.6. Pruebas de desarrollo de producto de sumas

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de desarrollo de producto de sumas
-- ==========================================

-- ------------------------------------------------------
-- Ej. 1. Sean a, b, c y d números enteros. Demostrar que
--    (a + b) * (c + d) = a * c + b * c + a * d + b * d
-- ------------------------------------------------------

import tactic
import data.int.basic

variables a b c d : ℤ

-- 1ª demostración
example : (a + b) * (c + d) = a * c + b * c + a * d + b * d :=
calc
  (a + b) * (c + d)
      = (a + b) * c + (a + b) * d         : by rw left_distrib
  ... = (a * c + b * c) + (a + b) * d     : by rw right_distrib
  ... = (a * c + b * c) + (a * d + b * d) : by rw right_distrib
  ... = a * c + b * c + a * d + b * d     : by rw ←add_assoc

-- 2ª demostración
example : (a + b) * (c + d) = a * c + b * c + a * d + b * d :=
by rw [left_distrib, right_distrib, right_distrib, ←add_assoc]

-- 3ª demostración
example : (a + b) * (c + d) = a * c + b * c + a * d + b * d :=
begin
  rw left_distrib,
  rw right_distrib,
  rw right_distrib,
  rw ←add_assoc,
end

-- 4ª demostración
example : (a + b) * (c + d) = a * c + b * c + a * d + b * d :=
calc
  (a + b) * (c + d)
      = (a + b) * c + (a + b) * d         : by rw mul_add
  ... = (a * c + b * c) + (a + b) * d     : by rw add_mul
```

```
  ... = (a * c + b * c) + (a * d + b * d) : by rw add_mul
  ... = a * c + b * c + a * d + b * d     : by rw ←add_assoc

-- 5ª demostración
example : (a + b) * (c + d) = a * c + b * c + a * d + b * d :=
-- by hint
by linarith

-- 6ª demostración
example : (a + b) * (c + d) = a * c + b * c + a * d + b * d :=
by nlinarith

-- 7ª demostración
example : (a + b) * (c + d) = a * c + b * c + a * d + b * d :=
by ring
```

Capítulo 4

Conjuntos

4.1. Elementos básicos sobre conjuntos

4.1.1. Pruebas de la reflexividad de la inclusión de conjuntos

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Prueba de la reflexividad de la inclusión de conjuntos
-- ======================================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    A ⊆ A
-- ----------------------------------------------------

import data.set
variable  U : Type
variable  x : U
variables A B C : set U

-- #reduce x ∈ A
-- #reduce B ⊆ C

-- 1ª demostración
example : A ⊆ A :=
begin
  intros x h,
  exact h,
end
```

```
-- 2ª demostración
example : A ⊆ A :=
assume x,
assume h : x ∈ A,
show x ∈ A, from h

-- 3ª demostración
example : A ⊆ A :=
assume x,
assume h : x ∈ A,
h

-- 4ª demostración
example : A ⊆ A :=
assume x,
λ h : x ∈ A, h

-- 5ª demostración
example : A ⊆ A :=
assume x,
id

-- 6ª demostración
example : A ⊆ A :=
λ x, id

-- 7ª demostración
example : A ⊆ A :=
-- by library_search
set.subset.rfl

open set

-- 8ª demostración
example : A ⊆ A :=
subset.rfl

-- 9ª demostración
example : A ⊆ A :=
-- by hint
by tauto

-- 10ª demostración
example : A ⊆ A :=
```

```
by finish

-- 11ª demostración
example : A ⊆ A :=
by refl
```

4.1.2. Pruebas de la antisimetría de la inclusión de conjuntos

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de la antisimetría de la inclusión de conjuntos
-- =======================================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    A ⊆ B, B ⊆ A ⊢ A = B
-- ----------------------------------------------------

import data.set

variable  U : Type
variables A B : set U

open set

-- 1ª demostración
example
  (h1 : A ⊆ B)
  (h2 : B ⊆ A)
  : A = B :=
begin
  ext,
  split,
  { intro h,
    exact h1 h, },
  { intro h,
    exact h2 h, },
end

-- 2ª demostración
example
```

```
  (h1 : A ⊆ B)
  (h2 : B ⊆ A)
  : A = B :=
ext
( assume x,
  iff.intro
  ( assume h : x ∈ A,
    show x ∈ B, from h1 h)
  ( assume h : x ∈ B,
    show x ∈ A, from h2 h))

-- 3ª demostración
example
  (h1 : A ⊆ B)
  (h2 : B ⊆ A)
  : A = B :=
ext
(λ x,
 iff.intro
 (λ h, h1 h)
 (λ h, h2 h))

-- 4ª demostración
example
  (h1 : A ⊆ B)
  (h2 : B ⊆ A)
  : A = B :=
eq_of_subset_of_subset
  ( assume x,
    assume h : x ∈ A,
    show x ∈ B, from h1 h)
  ( assume x,
    assume h : x ∈ B,
    show x ∈ A, from h2 h)

-- 5ª demostración
example
  (h1 : A ⊆ B)
  (h2 : B ⊆ A)
  : A = B :=
eq_of_subset_of_subset h1 h2

-- 6ª demostración
```

```
example
  (h1 : A ⊆ B)
  (h2 : B ⊆ A)
  : A = B :=
-- by library_search
subset.antisymm h1 h2
```

4.1.3. Introducción de la intersección

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Introducción de la intersección
-- ===============================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    x ∈ A → x ∈ B → x ∈ A ∩ B
-- ----------------------------------------------------

import data.set

variable  U : Type
variables A B : set U
variable  x : U

open set

-- #reduce x ∈ A ∩ B

-- 1ª demostración
example : x ∈ A → x ∈ B → x ∈ A ∩ B :=
begin
  intros h1 h2,
  simp,
  split,
  { exact h1, },
  { exact h2, },
end

-- 2ª demostración
example : x ∈ A → x ∈ B → x ∈ A ∩ B :=
begin
```

```
  intros h1 h2,
  split,
  { exact h1, },
  { exact h2, },
end

-- 3ª demostración
example : x ∈ A → x ∈ B → x ∈ A ∩ B :=
assume h1 : x ∈ A,
assume h2 : x ∈ B,
show x ∈ A ∩ B, from and.intro h1 h2

-- 4ª demostración
example : x ∈ A → x ∈ B → x ∈ A ∩ B :=
assume h1 : x ∈ A,
assume h2 : x ∈ B,
show x ∈ A ∩ B, from ⟨h1, h2⟩

-- 5ª demostración
example : x ∈ A → x ∈ B → x ∈ A ∩ B :=
assume h1 : x ∈ A,
assume h2 : x ∈ B,
⟨h1, h2⟩

-- 6ª demostración
example : x ∈ A → x ∈ B → x ∈ A ∩ B :=
λ h1 h2, ⟨h1, h2⟩

-- 7ª demostración
example : x ∈ A → x ∈ B → x ∈ A ∩ B :=
-- by library_search
mem_inter
```

4.1.4. Introducción de la unión

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla de introducción de la unión
-- =================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
```

```
--    A ⊆ A ∪ B
-- ----------------------------------------------------

import data.set

variable  U : Type
variables A B : set U
variable  x : U

open set

-- #reduce x ∈ A ∪ B

-- 1ª demostración
example : A ⊆ A ∪ B :=
begin
  intros x h,
  simp,
  left,
  exact h,
end

-- 2ª demostración
example : A ⊆ A ∪ B :=
begin
  intros x h,
  left,
  exact h,
end

-- 3ª demostración
example : A ⊆ A ∪ B :=
assume x,
assume h : x ∈ A,
show x ∈ A ∪ B, from or.inl h

-- 4ª demostración
example : A ⊆ A ∪ B :=
assume x,
assume h : x ∈ A,
or.inl h

-- 5ª demostración
example : A ⊆ A ∪ B :=
assume x,
```

```
λ h : x ∈ A, or.inl h

-- 6ª demostración
example : A ⊆ A ∪ B :=
assume x, or.inl

-- 7ª demostración
example : A ⊆ A ∪ B :=
λ x, or.inl

-- 8ª demostración
example : A ⊆ A ∪ B :=
-- by library_search
subset_union_left A B

-- 9ª demostración
example : A ⊆ A ∪ B :=
λ x, mem_union_left B

-- 10ª demostración
example : A ⊆ A ∪ B :=
-- by hint
by finish

-- 11ª demostración
example : A ⊆ A ∪ B :=
by simp
```

4.1.5. El conjunto vacío

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Regla del conjunto vacío
-- ========================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    ∅ ⊆ A
-- ----------------------------------------------------

import data.set
```

```
variable  U : Type
variables A : set U
variable  x : U

open set

-- #reduce (∅ : set U)
-- #reduce x ∈ (∅ : set U)

-- 1ª demostración
example : ∅ ⊆ A :=
begin
  intros x h,
  simp at h,
  exfalso,
  exact h,
end

-- 2ª demostración
example : ∅ ⊆ A :=
begin
  intros x h,
  exfalso,
  exact h,
end

-- 3ª demostración
example : ∅ ⊆ A :=
assume x,
assume h : x ∈ (∅ : set U),
show x ∈ A, from false.elim h

-- 4ª demostración
example : ∅ ⊆ A :=
λ x, λ h, false.elim h

-- 5ª demostración
example : ∅ ⊆ A :=
λ _, false.elim

-- 6ª demostración
example : ∅ ⊆ A :=
-- by library_search
empty_subset A
```

```
-- 7ª demostración
example : ∅ ⊆ A :=
assume x,
assume h : x ∈ (∅ : set U),
show x ∈ A, from absurd h (not_mem_empty x)

-- 8ª demostración
example : ∅ ⊆ A :=
λ x h, absurd h (not_mem_empty x)

-- 9ª demostración
example : ∅ ⊆ A :=
-- by hint
by tauto

-- 10ª demostración
example : ∅ ⊆ A :=
by finish

-- 11ª demostración
example : ∅ ⊆ A :=
by simp
```

4.1.6. Diferencia de conjuntos: $A \setminus B \subseteq A$

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Diferencia de conjuntos: A \ B ⊆ A
-- ==================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    A \ B ⊆ A
-- ----------------------------------------------------

import data.set

variable  U : Type
variables A B : set U
variable  x : U

open set
```

```
-- #reduce (A \ B)
-- #reduce x ∈ A \ B

-- 1ª demostración
example : A \ B ⊆ A :=
begin
  intros x h,
  simp at h,
  exact h.left,
end

-- 2ª demostración
example : A \ B ⊆ A :=
begin
  intros x h,
  exact h.left,
end

-- 3ª demostración
example : A \ B ⊆ A :=
assume x,
assume h : x ∈ A \ B,
show x ∈ A, from h.left

-- 4ª demostración
example : A \ B ⊆ A :=
assume x,
assume h : x ∈ A \ B,
and.left h

-- 5ª demostración
example : A \ B ⊆ A :=
assume x,
λ h, and.left h

-- 6ª demostración
example : A \ B ⊆ A :=
assume x, and.left

-- 7ª demostración
example : A \ B ⊆ A :=
λ _, and.left

-- 8ª demostración
```

```
example : A \ B ⊆ A :=
-- by library_search
diff_subset A B

-- 9ª demostración
example : A \ B ⊆ A :=
assume x,
assume h : x ∈ A \ B,
show x ∈ A, from mem_of_mem_diff h

-- 10ª demostración
example : A \ B ⊆ A :=
λ _, mem_of_mem_diff

-- 11ª demostración
example : A \ B ⊆ A :=
by finish [subset_def]
```

4.1.7. Complementario de un conjunto: Pruebas de $A \setminus B \subseteq B^c$

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Complementario de un conjunto: Pruebas de A \ B ⊆ Bᶜ
-- ====================================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    A \ B ⊆ Bᶜ
-- ----------------------------------------------------

import data.set

variable  {U : Type}
variables A B : set U
variable  x : U

open set

-- #reduce x ∈ Bᶜ
-- #reduce Bᶜ
```

```
-- 1ª demostración
example : A \ B ⊆ Bᶜ :=
begin
  intros x h,
  simp at *,
  exact h.right,
end

-- 2ª demostración
example : A \ B ⊆ Bᶜ :=
begin
  intros x h,
  exact h.right,
end

-- 3ª demostración
example : A \ B ⊆ Bᶜ :=
assume x,
assume h1 : x ∈ A \ B,
have h2 : x ∉ B, from and.right h1,
show x ∈ Bᶜ,     from h2

-- 4ª demostración
example : A \ B ⊆ Bᶜ :=
assume x,
assume h1 : x ∈ A \ B,
show x ∈ Bᶜ, from and.right h1

-- 5ª demostración
example : A \ B ⊆ Bᶜ :=
assume x,
λ h1, and.right h1

-- 6ª demostración
example : A \ B ⊆ Bᶜ :=
assume x,
and.right

-- 7ª demostración
example : A \ B ⊆ Bᶜ :=
λ _, and.right

-- 8ª demostración
example : A \ B ⊆ Bᶜ :=
```

```
λ _, not_mem_of_mem_diff
```

4.1.8. Pruebas de la conmutatividad de la intersección

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de la conmutatividad de la intersección
-- ===============================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    A ∩ B ⊆ B ∩ A
-- ----------------------------------------------------

import data.set

variable  {U : Type}
variables A B : set U
variable  x : U

open set

-- 1ª demostración
example : A ∩ B ⊆ B ∩ A :=
begin
  intros x h,
  simp at *,
  split,
  { exact h.right, },
  { exact h.left,  },
end

-- 2ª demostración
example : A ∩ B ⊆ B ∩ A :=
begin
  intros x h,
  split,
  { exact h.right, },
  { exact h.left,  },
end

-- 3ª demostración
example : A ∩ B ⊆ B ∩ A :=
```

```
begin
  rintros x ⟨h1, h2⟩,
  split,
  { exact h2, },
  { exact h1, },
end

-- 4ª demostración
example : A ∩ B ⊆ B ∩ A :=
begin
  rintros x ⟨h1, h2⟩,
  exact ⟨h2, h1⟩,
end

-- 5ª demostración
example : A ∩ B ⊆ B ∩ A :=
assume x,
assume h : x ∈ A ∩ B,
have h1 : x ∈ A, from and.left h,
have h2 : x ∈ B, from and.right h,
show x ∈ B ∩ A,  from and.intro h2 h1

-- 6ª demostración
example : A ∩ B ⊆ B ∩ A :=
assume x,
assume h : x ∈ A ∩ B,
have h1 : x ∈ A ∧ x ∈ B, from h,
have h2 : x ∈ B ∧ x ∈ A, from and.comm.mp h1,
show x ∈ B ∩ A,          from h2

-- 7ª demostración
example : A ∩ B ⊆ B ∩ A :=
assume x,
assume h : x ∈ A ∩ B,
show x ∈ B ∩ A, from and.comm.mp h

-- 8ª demostración
example : A ∩ B ⊆ B ∩ A :=
assume x,
assume h : x ∈ A ∩ B,
and.comm.mp h

-- 9ª demostración
example : A ∩ B ⊆ B ∩ A :=
```

```
assume x,
λ h, and.comm.mp h

-- 10ª demostración
example : A ∩ B ⊆ B ∩ A :=
assume x,
and.comm.mp

-- 10ª demostración
example : A ∩ B ⊆ B ∩ A :=
λ _, and.comm.mp

-- 11ª demostración
example : A ∩ B ⊆ B ∩ A :=
-- by hint
by finish

-- 12ª demostración
lemma aux : A ∩ B ⊆ B ∩ A :=
by simp

-- ------------------------------------------------------
-- Ej. 2. Demostrar
--    A ∩ B = B ∩ A
-- ------------------------------------------------------

-- 1ª demostración
example : A ∩ B = B ∩ A :=
begin
  apply eq_of_subset_of_subset,
  { exact aux A B, },
  { exact aux B A, },
end

-- 2ª demostración
example : A ∩ B = B ∩ A :=
eq_of_subset_of_subset (aux A B) (aux B A)

-- 3ª demostración
example : A ∩ B = B ∩ A :=
-- by library_search
inter_comm A B

-- 4ª demostración
example : A ∩ B = B ∩ A :=
```

```
-- by hint
by finish
```

4.2. Identidades conjuntistas

4.2.1. Pruebas de la propiedad distributiva de la intersección sobre la unión.

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)
-- ==========================================

import data.set
open set

variable  {U : Type}
variables A B C : set U

-- ------------------------------------------------------
-- Ej. 1. Demostrar
--    A ∩ (B ∪ C) ⊆ (A ∩ B) ∪ (A ∩ C)
-- ------------------------------------------------------

-- 1ª demostración
example :
  A ∩ (B ∪ C) ⊆ (A ∩ B) ∪ (A ∩ C) :=
begin
  intros x h,
  cases h with ha hbc,
  cases hbc with hb hc,
  { left,
    split,
    { exact ha, },
    { exact hb, }},
  { right,
    split,
    { exact ha, },
    { exact hc, }},
end
```

```
-- 2ª demostración
example :
  A ∩ (B ∪ C) ⊆ (A ∩ B) ∪ (A ∩ C) :=
begin
  intros x h,
  cases h with ha hbc,
  cases hbc with hb hc,
  { left,
    split,
    { assumption, },
    { assumption, }},
  { right,
    split,
    { assumption, },
    { assumption, }},
end

-- 3ª demostración
example :
  A ∩ (B ∪ C) ⊆ (A ∩ B) ∪ (A ∩ C) :=
begin
  intros x h,
  cases h with ha hbc,
  cases hbc with hb hc,
  { left,
    split,
    assumption', },
  { right,
    split,
    assumption', },
end

-- 4ª demostración
example :
  A ∩ (B ∪ C) ⊆ (A ∩ B) ∪ (A ∩ C) :=
begin
  rintros x ⟨ha, (hb | hc)⟩,
  { left,
    split,
    assumption', },
  { right,
    split,
    assumption', },
end
```

```
-- 5ª demostración
example :
  A ∩ (B ∪ C) ⊆ (A ∩ B) ∪ (A ∩ C) :=
assume x,
assume h : x ∈ A ∩ (B ∪ C),
have x ∈ A, from and.left h,
have x ∈ B ∪ C, from and.right h,
or.elim (‹x ∈ B ∪ C›)
  ( assume : x ∈ B,
    have x ∈ A ∩ B, from and.intro ‹x ∈ A› ‹x ∈ B›,
    show x ∈ (A ∩ B) ∪ (A ∩ C), from or.inl this)
  ( assume : x ∈ C,
    have x ∈ A ∩ C, from and.intro ‹x ∈ A› ‹x ∈ C›,
    show x ∈ (A ∩ B) ∪ (A ∩ C), from or.inr this)

-- 6ª demostración
lemma inter_union_l1 :
  A ∩ (B ∪ C) ⊆ (A ∩ B) ∪ (A ∩ C) :=
assume x,
assume h : x ∈ A ∩ (B ∪ C),
have ha : x ∈ A, from and.left h,
have hbc : x ∈ B ∪ C, from and.right h,
or.elim hbc
  ( assume hb : x ∈ B,
    have hab: x ∈ A ∩ B, from and.intro ha hb,
    show x ∈ (A ∩ B) ∪ (A ∩ C), from or.inl hab)
  ( assume hc : x ∈ C,
    have hac : x ∈ A ∩ C, from and.intro ha hc,
    show x ∈ (A ∩ B) ∪ (A ∩ C), from or.inr hac)

-- ----------------------------------------------------
-- Ej. 2. Demostrar
--    (A ∩ B) ∪ (A ∩ C) ⊆ A ∩ (B ∪ C)
-- ----------------------------------------------------

-- 1ª demostración
example :
  (A ∩ B) ∪ (A ∩ C) ⊆ A ∩ (B ∪ C) :=
begin
  intros x h,
  cases h with hab hac,
  { split,
    { exact hab.left, },
```

```
    { left,
      exact hab.right, }},
  { split,
    { exact hac.left, },
    { right,
      exact hac.right, }},
end

-- 2ª demostración
example :
  (A ∩ B) ∪ (A ∩ C) ⊆ A ∩ (B ∪ C) :=
begin
  rintros x (⟨ha, hb⟩ | ⟨ha, hc⟩),
  { split,
    { exact ha, },
    { left,
      exact hb, }},
  { split,
    { exact ha, },
    { right,
      exact hc, }},
end

-- 3ª demostración
lemma inter_union_l2 :
  (A ∩ B) ∪ (A ∩ C) ⊆ A ∩ (B ∪ C) :=
assume x,
assume : x ∈ (A ∩ B) ∪ (A ∩ C),
or.elim this
  ( assume h : x ∈ A ∩ B,
    have x ∈ A, from and.left h,
    have x ∈ B, from and.right h,
    have x ∈ B ∪ C, from or.inl this,
    show x ∈ A ∩ (B ∪ C), from and.intro ‹x ∈ A› this)
  ( assume h : x ∈ A ∩ C,
    have x ∈ A, from and.left h,
    have x ∈ C, from and.right h,
    have x ∈ B ∪ C, from or.inr this,
    show x ∈ A ∩ (B ∪ C), from and.intro ‹x ∈ A› this)

-- ----------------------------------------------------
-- Ej. 3. Demostrar
--    (A ∩ B) ∪ (A ∩ C) = A ∩ (B ∪ C)
-- ----------------------------------------------------
```

```
-- 1ª demostración
example :
  A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C) :=
-- by library_search
inter_distrib_left A B C

-- 2ª demostración
theorem inter_union :
  A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C) :=
eq_of_subset_of_subset
  (inter_union_l1 A B C)
  (inter_union_l2 A B C)

-- 3ª demostración
example :
  A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C) :=
begin
  ext,
  simp,
  exact and_or_distrib_left,
end

-- 4ª demostración
example :
  A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C) :=
begin
  ext,
  exact and_or_distrib_left,
end

-- 5ª demostración
example :
  A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C) :=
ext (λ x, and_or_distrib_left)
```

4.2.2. Pruebas de (A ∩ Bᶜ) ∪ B = A ∪ B

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de (A ∩ Bᶜ) ∪ B = A ∪ B
-- ================================
```

```
-- ----------------------------------------------------
-- Ej. 1. Demostrar
--      (A ∩ Bᶜ) ∪ B = A ∪ B
-- ----------------------------------------------------

import data.set

open set

variable  U : Type
variables A B C : set U

-- 1ª demostración
-- ===============

example : (A ∩ Bᶜ) ∪ B = A ∪ B :=
calc
  (A ∩ Bᶜ) ∪ B = (A ∪ B) ∩ (Bᶜ ∪ B) : by rw union_distrib_right
           ... = (A ∪ B) ∩ univ     : by rw compl_union_self
           ... = A ∪ B              : by rw inter_univ

example : (A ∩ B) ∪ C = (A ∪ C) ∩ (B ∪ C) :=
-- by library_search
union_distrib_right A B C

example : Bᶜ ∪ B = univ :=
-- by library_search
compl_union_self B

example : A ∩ univ = A :=
-- by library_search
inter_univ A

-- 2ª demostración
-- ===============

example : (A ∩ Bᶜ) ∪ B = A ∪ B :=
begin
  rw union_distrib_right,
  rw compl_union_self,
  rw inter_univ,
end

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

```
example : (A ∩ Bᶜ) ∪ B = A ∪ B :=
by rw [union_distrib_right, compl_union_self, inter_univ]

-- 4ª demostración
-- ===============

example : (A ∩ Bᶜ) ∪ B = A ∪ B :=
by simp [union_distrib_right]
```

4.3. Familias de conjuntos

4.3.1. Unión e intersección de familias de conjuntos

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Unión e intersección de familias de conjuntos
-- =============================================

-- ----------------------------------------------------
-- Ej. 1. Declarar I y U como variables de tipo.
-- ----------------------------------------------------

variables {I U : Type}

-- ----------------------------------------------------
-- Ej. 2. Definir la función
--    Union : (I → set U) → set U
-- tal que (Union A) es la unión de de los conjuntos de
-- la familia A.
-- ----------------------------------------------------

def Union (A : I → set U) : set U :=
  { x | ∃ i : I, x ∈ A i }

-- ----------------------------------------------------
-- Ej. 3. Definir la función
--    Inter : (I → set U) → set U
-- tal que (Inter A) es la intersección de de los
-- conjuntos de la familia A.
-- ----------------------------------------------------
```

```
def Inter (A : I → set U) : set U :=
  { x | ∀ i : I, x ∈ A i }

-- ----------------------------------------------------
-- Ej. 4. Declarar
-- + x como una variable sobre U y
-- + A como una variable sobre familas de conjuntos de
--   U con índice en A.
-- ----------------------------------------------------

variable x : U
variable (A : I → set U)

-- ----------------------------------------------------
-- Ej. 5. Demostrar que
--    x ∈ Union A ⊢ ∃ i, x ∈ A i
-- ----------------------------------------------------

example
  (h : x ∈ Union A)
  : ∃ i, x ∈ A i :=
h

-- ----------------------------------------------------
-- Ej. 6. Demostrar que
--    x ∈ x ∈ Inter A ⊢ ∀ i, x ∈ A i
-- ----------------------------------------------------

example
  (h : x ∈ Inter A)
  : ∀ i, x ∈ A i :=
h

-- ----------------------------------------------------
-- Ej 7. Usar (⋃ i, A i) como notación para (Union A).
-- ----------------------------------------------------

notation `⋃` binders `, ` r:(scoped f, Union f) := r

-- ----------------------------------------------------
-- Ej 8. Usar (⋂ i, A i) como notación para (Inter A).
-- ----------------------------------------------------

notation `⋂` binders `, ` r:(scoped f, Inter f) := r
```

```
-- ----------------------------------------------------
-- Ej. 9. Demostrar que
--    x ∈ ⋃ i, A i ⊢ ∃ i, x ∈ A i
-- ----------------------------------------------------

example
  (h : x ∈ ⋃ i, A i)
  : ∃ i, x ∈ A i :=
h

-- ----------------------------------------------------
-- Ej. 10. Demostrar que
--    x ∈ x ∈ ⋂ i, A i ⊢ ∀ i, x ∈ A i
-- ----------------------------------------------------

example
  (h : x ∈ ⋂ i, A i)
  : ∀ i, x ∈ A i :=
h
```

4.3.2. Pertenencia a uniones e intersecciones de familias

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pertenencia a uniones e intersecciones de familias
-- ===================================================

import data.set

open set

variables {I U : Type}
variables {A : I → set U}
variable  {x : U}

-- ----------------------------------------------------
-- Ej. 1. Demostrar que
--    (x ∈ ⋃ i, A i) ↔ (∃ i, x ∈ A i)
-- ----------------------------------------------------
```

```
-- 1ª demostración
example :
  (x ∈ ⋃ i, A i) ↔ (∃ i, x ∈ A i) :=
-- by library_search
mem_Union

-- 2ª demostración
example :
  (x ∈ ⋃ i, A i) ↔ (∃ i, x ∈ A i) :=
by simp

-- ----------------------------------------------------
-- Ej. 2. Demostrar que
--    (x ∈ ⋂ i, A i) ↔ (∀ i, x ∈ A i)
-- ----------------------------------------------------

-- 1ª demostración
example :
  (x ∈ ⋂ i, A i) ↔ (∀ i, x ∈ A i) :=
-- by library_search
mem_Inter

-- 2ª demostración
example :
  (x ∈ ⋂ i, A i) ↔ (∀ i, x ∈ A i) :=
by simp
```

4.3.3. Pruebas de la distributiva de la intersección general sobre la intersección

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de la distributiva de la intersección general sobre la intersección
-- ==========================================================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    (⋂ i, A i ∩ B i) = (⋂ i, A i) ∩ (⋂ i, B i)
-- ----------------------------------------------------

import data.set
import tactic
```

```
open set

variables {I U : Type}
variables {A B : I → set U}

-- 1ª demostración
example :
  (⋂ i, A i ∩ B i) = (⋂ i, A i) ∩ (⋂ i, B i) :=
begin
  ext,
  split,
  { intro h,
    rw mem_Inter at h,
    split,
    { rw mem_Inter,
      intro i,
      exact (h i).left, },
    { rw mem_Inter,
      intro i,
      exact (h i).right, }},
  { rintro ⟨h1, h2⟩,
    rw mem_Inter at *,
    intro i,
    exact ⟨h1 i, h2 i⟩, },
end

-- 2ª demostración
example :
  (⋂ i, A i ∩ B i) = (⋂ i, A i) ∩ (⋂ i, B i) :=
ext $
assume x : U,
iff.intro
( assume h : x ∈ ⋂ i, A i ∩ B i,
  have h1 : ∀ i, x ∈ A i ∩ B i,
    from mem_Inter.mp h,
  have h2 : ∀ i, x ∈ A i,
    from assume i, and.left (h1 i),
  have h3 : ∀ i, x ∈ B i,
    from assume i, and.right (h1 i),
  have h4 : x ∈ ⋂ i, A i,
    from mem_Inter.mpr h2,
  have h5 : x ∈ ⋂ i, B i,
    from mem_Inter.mpr h3,
  show x ∈ (⋂ i, A i) ∩ (⋂ i, B i),
```

```
    from and.intro h4 h5)
( assume h : x ∈ (⋂ i, A i) ∩ (⋂ i, B i),
  have h1 : ∀ i, x ∈ A i,
    from mem_Inter.mp (and.left h),
  have h2 : ∀ i, x ∈ B i,
    from mem_Inter.mp (and.right h),
  have h3 : ∀ i, x ∈ A i ∩ B i,
    from assume i, and.intro (h1 i) (h2 i),
  show x ∈ ⋂ i, A i ∩ B i,
    from mem_Inter.mpr h3)

-- 3ª demostración
example :
  (⋂ i, A i ∩ B i) = (⋂ i, A i) ∩ (⋂ i, B i) :=
-- by library_search
Inter_inter_distrib A B

-- 4ª demostración
example :
  (⋂ i, A i ∩ B i) = (⋂ i, A i) ∩ (⋂ i, B i) :=
ext (by finish)
```

4.3.4. Reglas de la intersección general

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Reglas de la intersección general
-- ==================================

import data.set
open set

section
variables {I U : Type}
variables {A : I → set U}
variable  {x : U}

-- Regla de introducción de la intersección
-- ========================================

-- 1ª demostración
example
```

```
  (h : ∀ i, x ∈ A i)
  : x ∈ ⋂ i, A i :=
begin
  simp,
  assumption,
end

-- 2ª demostración
theorem Inter.intro
  (h : ∀ i, x ∈ A i)
  : x ∈ ⋂ i, A i :=
by simp; assumption

-- Regla de eliminación de la intersección
-- ========================================

-- 1ª demostración
example
  (h : x ∈ ⋂ i, A i)
  (i : I)
  : x ∈ A i :=
begin
  simp at h,
  apply h,
end

-- 2ª demostración
@[elab_simple]
theorem Inter.elim
  (h : x ∈ ⋂ i, A i)
  (i : I)
  : x ∈ A i :=
by simp at h; apply h

end
```

4.3.5. Reglas de la unión general

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Reglas de la unión general
-- ==========================
```

```
import data.set

open set

variables {I U : Type}
variables {A : I → set U}
variable  {x : U}
variable  (i : I)

-- Regla de introducción de la unión
-- ==================================

-- 1ª demostración
example
  (h : x ∈ A i)
  : x ∈ ⋃ i, A i :=
begin
  simp,
  existsi i,
  exact h
end

-- 2ª demostración
theorem Union.intro
  (h : x ∈ A i)
  : x ∈ ⋃ i, A i :=
by {simp, existsi i, exact h}

-- Regla de eliminación de la unión
-- ================================

-- 1ª demostración
example
  {b : Prop}
  (h₁ : x ∈ ⋃ i, A i)
  (h₂ : ∀ (i : I), x ∈ A i → b)
  : b :=
begin
  simp at h₁,
  cases h₁ with i h,
  exact h₂ i h,
end

-- 2ª demostración
```

```
theorem Union.elim
  {b : Prop}
  (h₁ : x ∈ ⋃ i, A i)
  (h₂ : ∀ (i : I), x ∈ A i → b)
  : b :=
by {simp at h₁, cases h₁ with i h, exact h₂ i h}
```

4.3.6. Pruebas de intersección sobre unión general

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de intersección sobre unión general
-- ===========================================

import data.set

open set

variables {I U : Type}
variables {A : I → set U}
variable  {C : set U}

-- ------------------------------------------------------
-- Ej. 1. Demostrar
--    C ∩ (⋃i, A i) ⊆ (⋃ i, C ∩ A i)
-- ------------------------------------------------------

-- 1ª demostración
example :
  C ∩ (⋃i, A i) ⊆ (⋃ i, C ∩ A i)  :=
begin
  rintros x ⟨hC, hU⟩,
  rw mem_Union at hU,
  cases hU with i hA,
  apply mem_Union.mpr,
  use i,
  split,
  assumption',
end

-- 2ª demostración
example :
```

```
  C ∩ (⋃i, A i) ⊆ (⋃ i, C ∩ A i)  :=
begin
  intros x h,
  simp * at *,
end

-- 3ª demostración
lemma inter_Uni_l1 :
  C ∩ (⋃i, A i) ⊆ (⋃ i, C ∩ A i)  :=
by {intros x h, simp * at *}

-- ---------------------------------------------------
-- Ej. 2. Demostrar
--    (⋃ i, C ∩ A i) ⊆ C ∩ (⋃i, A i)
-- ---------------------------------------------------

-- 1ª demostración
example :
  (⋃ i, C ∩ A i) ⊆ C ∩ (⋃i, A i) :=
begin
  intros x h,
  rw mem_Union at h,
  cases h with i hi,
  cases hi with hC hA,
  split,
  { exact hC, },
  { apply mem_Union.mpr,
    use i,
    exact hA, },
end

-- 2ª demostración
example : (⋃ i, C ∩ A i) ⊆ C ∩ (⋃i, A i) :=
begin
  intros x h,
  rw mem_Union at h,
  rcases h with ⟨i, hC, hA⟩,
  split,
  { exact hC, },
  { apply mem_Union.mpr,
    use i,
    exact hA, },
end

-- 3ª demostración
```

```
example :
  (⋃ i, C ∩ A i) ⊆ C ∩ (⋃i, A i) :=
begin
  intros x h,
  simp * at *,
end

-- 4ª demostración
lemma inter_Uni_l2 :
  (⋃ i, C ∩ A i) ⊆ C ∩ (⋃i, A i) :=
by {intros x h, simp * at *}

-- ---------------------------------------------------
-- Ej. 3. Demostrar
--    C ∩ (⋃i, A i) = (⋃ i, C ∩ A i)
-- ---------------------------------------------------

-- 1ª demostración
example :
  C ∩ (⋃i, A i) = (⋃ i, C ∩ A i) :=
eq_of_subset_of_subset inter_Uni_l1 inter_Uni_l2

-- 2ª demostración
example :
  C ∩ (⋃i, A i) = (⋃ i, C ∩ A i) :=
-- by library_search
inter_Union C A

-- 3ª demostración
example :
  C ∩ (⋃i, A i) = (⋃ i, C ∩ A i) :=
ext $ by simp

-- 4ª demostración
example :
  C ∩ (⋃i, A i) = (⋃ i, C ∩ A i) :=
by {ext, simp}
```

4.3.7. Pruebas de (⋃i, ⋂j, A i j) ⊆ (⋂j, ⋃i, A i j)

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de (⋃i, ⋂j, A i j) ⊆ (⋂j, ⋃i, A i j)
-- ============================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar
--    (⋃i, ⋂j, A i j) ⊆ (⋂j, ⋃i, A i j)
-- ----------------------------------------------------

import data.set

open set

variables {I J U : Type}
variables (A : I → J → set U)

-- 1ª demostración
example : (⋃i, ⋂j, A i j) ⊆ (⋂j, ⋃i, A i j) :=
begin
  intros x h,
  rw mem_Union at h,
  cases h with i hi,
  rw mem_Inter at hi,
  apply mem_Inter.mpr,
  intro j,
  apply mem_Union.mpr,
  use i,
  exact (hi j),
end

-- 2ª demostración
example : (⋃i, ⋂j, A i j) ⊆ (⋂j, ⋃i, A i j) :=
begin
  intros x h,
  simp * at *,
  cases h with i hi,
  intro j,
  use i,
  exact (hi j),
end
```

4.4. Conjunto potencia

4.4.1. Definición del conjunto potencia

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Definición del conjunto potencia
-- ================================

variable {U : Type}

def powerset (A : set U) : set (set U) :=
  {B : set U | B ⊆ A}

example
  (A B : set U)
  (h : B ∈ powerset A)
  : B ⊆ A :=
h
```

4.4.2. Pruebas de A ∈ 𝒫 (A ∪ B)

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Pruebas de A ∈ 𝒫 (A ∪ B)
-- ========================

import data.set
open set

variable  {U : Type}
variables (A B : set U)

#reduce powerset A
#reduce B ∈ powerset A
#reduce 𝒫 A
#reduce B ∈ 𝒫 A

-- ?ª demostración
example : A ∈ 𝒫 (A ∪ B) :=
begin
```

```
  intros x h,
  simp,
  left,
  exact h,
end

-- ?ª demostración
example : A ⊆ 𝒫 (A ∪ B) :=
begin
  intros x h,
  exact or.inl h,
end

-- ?ª demostración
example : A ⊆ 𝒫 (A ∪ B) :=
λ x, or.inl

-- ?ª demostración
example : A ⊆ 𝒫 (A ∪ B) :=
assume x,
assume : x ∈ A,
show x ∈ A ∪ B, from or.inl ‹x ∈ A›
```

4.4.3. Monotonía del conjunto potencia: 𝒫 A ⊆ 𝒫 B ↔ A ⊆ B

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Monotonía del conjunto potencia: 𝒫 A ⊆ 𝒫 B ↔ A ⊆ B
-- ====================================================

import data.set
open set

variable  {U : Type}
variables {A B C : set U}

-- #reduce 𝒫 A
-- #reduce B ∈ 𝒫 A

-- ----------------------------------------------------
-- Ej. 1. Demostrar
```

```
--    𝒫 A ⊆ 𝒫 B → A ⊆ B
-- -------------------------------------------------------

-- 1ª demostración
example : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
begin
  intro h,
  apply subset_of_mem_powerset,
  apply h,
  apply mem_powerset,
  exact subset.rfl,
end

-- 2ª demostración
example : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
begin
  intro h,
  apply h,
  exact subset.rfl,
end

-- 3ª demostración
example : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
begin
  intro h,
  exact (h subset.rfl),
end

-- 4ª demostración
example : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
λ h, h subset.rfl

-- 5ª demostración
example : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
assume h1 : 𝒫 A ⊆ 𝒫 B,
have h2 : A ⊆ A, from subset.rfl,
have h3 : A ∈ 𝒫 A, from h2,
have h4 : A ∈ 𝒫 B, from h1 h3,
show A ⊆ B, from h4

-- 6ª demostración
example : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
assume h1 : 𝒫 A ⊆ 𝒫 B,
have h2 : A ⊆ A, from subset.rfl,
```

```
have h3 : A ∈ 𝒫 A, from h2,
h1 h3

-- 7ª demostración
example : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
assume h1 : 𝒫 A ⊆ 𝒫 B,
have h2 : A ⊆ A, from subset.rfl,
h1 h2

-- 8ª demostración
example : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
assume h1 : 𝒫 A ⊆ 𝒫 B,
h1 subset.rfl

-- 9ª demostración
lemma aux1 : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
λ h, h subset.rfl

-- 10ª demostración
example : 𝒫 A ⊆ 𝒫 B → A ⊆ B :=
powerset_mono.mp

-- ----------------------------------------------------
-- Ej. 2. Demostrar
--    A ⊆ B → 𝒫 A ⊆ 𝒫 B
-- ----------------------------------------------------

-- 1ª demostración
example : A ⊆ B → 𝒫 A ⊆ 𝒫 B :=
begin
  intro h,
  intros C hCA,
  apply mem_powerset,
  apply subset.trans hCA h,
end

-- 2ª demostración
example : A ⊆ B → 𝒫 A ⊆ 𝒫 B :=
begin
  intros h C hCA,
  apply subset.trans hCA h,
end

-- 3ª demostración
```

```
lemma aux2 : A ⊆ B → 𝒫 A ⊆ 𝒫 B :=
λ h C hCA, subset.trans hCA h

-- 4ª demostración
example : A ⊆ B → 𝒫 A ⊆ 𝒫 B :=
powerset_mono.mpr

-- ----------------------------------------------------
-- Ej. 3. Demostrar
--    𝒫 A ⊆ 𝒫 B ↔ A ⊆ B
-- ----------------------------------------------------

-- 1ª demostración
example : 𝒫 A ⊆ 𝒫 B ↔ A ⊆ B :=
iff.intro aux1 aux2

-- 2ª demostración
example : 𝒫 A ⊆ 𝒫 B ↔ A ⊆ B :=
-- by library_search
powerset_mono

-- 3ª demostración
example : 𝒫 A ⊆ 𝒫 B ↔ A ⊆ B :=
-- by hint
by finish

-- 4ª demostración
example : 𝒫 A ⊆ 𝒫 B ↔ A ⊆ B :=
by simp
```

Capítulo 5

Relaciones

5.1. Relaciones de orden

5.1.1. Las irreflexivas y transitivas son asimétricas

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Las irreflexivas y transitivas son asimétricas
-- ==============================================

-- ----------------------------------------------------
-- Ej. 1. Demostrar que las relaciones irreflexivas y
-- transitivas son asimétricas.
-- ----------------------------------------------------

variable A : Type
variable R : A → A → Prop

-- #reduce irreflexive R
-- #reduce transitive R

-- 1ª demostración
example
  (h1 : irreflexive R)
  (h2 : transitive R)
  : ∀ x y, R x y → ¬ R y x :=
begin
  intros x y h3 h4,
  apply h1 x,
  apply h2 h3 h4,
end
```

```
-- 2ª demostración
example
  (h1 : irreflexive R)
  (h2 : transitive R)
  : ∀ x y, R x y → ¬ R y x :=
begin
  intros x y h3 h4,
  apply (h1 x) (h2 h3 h4),
end

-- 3ª demostración
example
  (h1 : irreflexive R)
  (h2 : transitive R)
  : ∀ x y, R x y → ¬ R y x :=
λ x y h3 h4, (h1 x) (h2 h3 h4)

-- 4ª demostración
example
  (h1 : irreflexive R)
  (h2 : transitive R)
  : ∀ x y, R x y → ¬ R y x :=
assume x y,
assume h3 : R x y,
assume h4 : R y x,
have h5 : R x x, from h2 h3 h4,
have h6 : ¬ R x x, from h1 x,
show false, from h6 h5
```

5.1.2. Las partes estrictas son irreflexivas

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Las partes estrictas son irreflexivas
-- =====================================

-- ----------------------------------------------------
-- Ej. 1. La parte estricta de una relación R es la
-- relación R' definida por
--    R' a b := R a b ∧ a ≠ b
--
-- Demostrar que la parte estricta de cualquier
```

```
-- relación es irreflexiva.
-- ----------------------------------------------------

import tactic

section

parameter {A : Type}
parameter (R : A → A → Prop)

definition R' (a b : A) : Prop :=
  R a b ∧ a ≠ b

#reduce irreflexive R

-- 1ª demostración
example :
  irreflexive R' :=
begin
  intros a h,
  cases h with h1 h2,
  apply h2,
  refl,
end

-- 2ª demostración
example :
  irreflexive R' :=
assume a,
assume : R' a a,
have a ≠ a, from and.right this,
have a = a, from rfl,
show false, from ‹a ≠ a› ‹a = a›

end
```

5.1.3. Las partes estrictas de los órdenes parciales son transitivas

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Las partes estrictas de los órdenes parciales son transitivas
-- ==============================================================

-- ----------------------------------------------------
-- Ej. 1. La parte estricta de una relación R es la
-- relación R' definida por
--    R' x y := R x y ∧ R y x
--
-- Demostrar que si R es un orden parcial, entonces su
-- parte estricta es reflexiva.
-- ----------------------------------------------------

import tactic

section

parameter {A : Type}
parameter (R : A → A → Prop)
parameter (reflR    : reflexive R)
parameter (transR   : transitive R)
parameter (antisimR : anti_symmetric R)
variables {a b c : A}

definition R' (a b : A) : Prop :=
  R a b ∧ a ≠ b

include transR
include antisimR

-- 1ª demostración
example : R' a b → R' b c → R' a c :=
begin
  rintros ⟨h1,h2⟩ ⟨h3,h4⟩,
  split,
  { apply (transR h1 h3), },
  { intro h5,
    apply h4,
    apply (antisimR h3),
    rw ←h5,
    exact h1, },
end

-- 2ª demostración
-- ===============
```

```
local infix ≤ := R
local infix < := R'

example
  (h₁ : a < b)
  (h₂ : b < c)
  : a < c :=
have a ≤ b, from and.left h₁,
have a ≠ b, from and.right h₁,
have b ≤ c, from and.left h₂,
have b ≠ c, from and.right h₂,
have a ≤ c, from transR ‹a ≤ b› ‹b ≤ c›,
have a ≠ c, from
    assume : a = c,
    have c ≤ b, from eq.subst ‹a = c› ‹a ≤ b›,
    have b = c, from antisimR ‹b ≤ c› ‹c ≤ b›,
    show false, from ‹b ≠ c› ‹b = c›,
show a < c, from and.intro ‹a ≤ c› ‹a ≠ c›

end
```

5.1.4. Las partes simétricas de las reflexivas son reflexivas

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Las partes simétricas de las reflexivas son reflexivas
-- ======================================================

-- ----------------------------------------------------
-- Ej. 1. La parte simétrica de una relación R es la
-- relación S definida por
--    S x y := R x y ∧ R y x
--
-- Demostrar que la parte simétrica de una relación
-- reflexiva es reflexiva.
-- ----------------------------------------------------

section

parameter A : Type
parameter R : A → A → Prop
```

```
parameter reflR : reflexive R

include reflR

def S (x y : A) := R x y ∧ R y x

-- 1ª demostración
example : reflexive S :=
begin
  intro x,
  split,
  { exact (reflR x), },
  { exact (reflR x), },
end

-- 2ª demostración
example : reflexive S :=
assume x,
have R x x, from reflR x,
show S x x, from and.intro this this

end
```

5.1.5. Las partes simétricas son simétricas

- Enlaces al código, a la sesión en Lean Web y al vídeo.

```
-- Las partes simétricas son transitivas
-- =====================================

-- ----------------------------------------------------
-- Ej. 1. La parte simétrica de una relación R es la
-- relación S definida por
--    S x y := R x y ∧ R y x
--
-- Demostrar que la parte simétrica de cualquier
-- relación es transitiva.
-- ----------------------------------------------------

section
parameter A : Type
parameter R : A → A → Prop
```

```
def S (x y : A) := R x y ∧ R y x

-- 1ª demostración
example : symmetric S :=
begin
  intros x y h,
  split,
  { exact h.right, },
  { exact h.left, },
end

-- 2ª demostración
example : symmetric S :=
assume x y,
assume h : S x y,
have h1 : R x y, from h.left,
have h2 : R y x, from h.right,
show S y x, from ⟨h2, h1⟩

-- 3ª demostración
example : symmetric S :=
assume x y,
assume h : S x y,
show S y x, from ⟨h.right, h.left⟩

-- 4ª demostración
example : symmetric S :=
λ x y h, ⟨h.right, h.left⟩

end
```