Lógica con Lean

José A. Alonso Jiménez

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, 29 de septiembre 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 → Q, Q → R ⊢ P → 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 :=
begin
  split,
  { cases hPQ with hP hQ,
    clear hP,
    exact hQ, },
  { exact hR, },
```

```
end

-- 8ª demostración
-- ===============

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

-- 9ª demostración
-- ===============

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

-- 10ª 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 :=
or.rec h2 h3 h1

-- 3ª 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

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

-- 5ª 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

-- 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 :=
by tauto

-- 9ª 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))

-- 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 :=
by tauto
```

```
-- 10ª 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

-- 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 :=
by tauto

-- 12ª 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⟩)

-- 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. Demostraciones por reducción al absurdo (o por contradicción)

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
example
  (h1 : ¬P → false)
  : P :=
by finish
```

2.6.2. 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 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
open_locale classical

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 (λh1, h1 (or.inr (λh2, h1 (or.inl h2))))

-- 16ª demostración
example : F ∨ ¬F :=
by tauto

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

2.6.3. 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

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

```
begin
  by_contra h2,
  exact h1 h2,
end

-- 7ª demostración
example
  (h1 : ¬¬P)
  : P :=
by tauto

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