Ejercicios de lógica de primer orden con Lean

José A. Alonso Jiménez

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, 12 de diciembre de 2020

Índice general

Capítulo 1

Introducción

Este libro es una colección de soluciones de ejercicios de lógica de primer orden (LPO) formalizadas con Lean que complementa el libro de Lógica con Lean y es continuación del libro Ejercicios de lógica proposicional con Lean.

Para cada uno de los ejercicios se formalizan las soluciones en distintos estilos:

- aplicativo usando tácticas con razonamiento hacia atrás,
- declarativo (o estructurado) con razonamiento hacia adelante,
- funcional con términos del tipo especificado y
- automático.

Las demostraciones funcionales se obtienen mediante una sucesión de transformaciones de una aplicativa (o declarativa) eliminando elementos no esenciales.

Además, al final de cada ejercicio se encuentra un enlace al código y otro a una sesión de Lean en la Web que contiene la solución del ejercicio.

Capítulo 2

Ejercicios sobre cuantificadores

2.1. $\forall x, P\ x \longrightarrow Q\ x \vdash (\forall x, P\ x) \longrightarrow (\forall x, Q\ x)$

```
-- ----------------------------------------------------
-- Ejercicio 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
  (h : ∀ x, P x → Q x)
  : (∀ x, P x) → (∀ x, Q x) :=
begin
  intros h1 a,
  exact (h a) (h1 a),
end

-- 2ª demostración
example
  (h : ∀ x, P x → Q x)
  : (∀ x, P x) → (∀ x, Q x) :=
λ h1 a, (h a) (h1 a)

-- 3ª demostración
example
  (h : ∀ x, P x → Q x)
```

```
  : (∀ x, P x) → (∀ x, Q x) :=
assume h1 : ∀ x, P x,
show ∀ x, Q x, from
  assume a,
  have h2 : P a → Q a,
    from h a,
  have h3 : P a,
    from h1 a,
  show Q a,
    from h2 h3

-- 4ª demostración
example
  (h : ∀ x, P x → Q x)
  : (∀ x, P x) → (∀ x, Q x) :=
assume h1 : ∀ x, P x,
show ∀ x, Q x, from
  assume a,
  have h2 : P a → Q a,
    from h a,
  have h3 : P a,
    from h1 a,
  h2 h3

-- 5ª demostración
example
  (h : ∀ x, P x → Q x)
  : (∀ x, P x) → (∀ x, Q x) :=
assume h1 : ∀ x, P x,
show ∀ x, Q x, from
  assume a,
  (h a) (h1 a)

-- 6ª demostración
example
  (h : ∀ x, P x → Q x)
  : (∀ x, P x) → (∀ x, Q x) :=
assume h1 : ∀ x, P x,
λ a, (h a) (h1 a)

-- 7ª demostración
example
  (h : ∀ x, P x → Q x)
  : (∀ x, P x) → (∀ x, Q x) :=
λ h1 a, (h a) (h1 a)
```

```
-- 8ª demostración
example
  (h : ∀ x, P x → Q x)
  : (∀ x, P x) → (∀ x, Q x) :=
-- by library_search
forall_imp h

-- 9ª demostración
example
  (h : ∀ x, P x → Q x)
  : (∀ x, P x) → (∀ x, Q x) :=
-- by hint
by tauto

-- 10ª demostración
example
  (h : ∀ x, P x → Q x)
  : (∀ x, P x) → (∀ x, Q x) :=
by finish
```

Enlaces al código y a la sesión en Lean Web.

2.1.1. ∃ x, ¬(P x) ⊢ ¬(∀ x, P x)

Enlaces al código y a la sesión en Lean Web.

```
-- ----------------------------------------------------
-- Ejercicio 2. Demostrar
--    ∃ x, ¬(P x) ⊢ ¬(∀ x, P x)
-- ----------------------------------------------------

import tactic

variable (U : Type)
variable (P : U -> Prop)

-- 1ª demostración
example
  (h: ∃ x, ¬(P x))
  : ¬(∀ x, P x) :=
begin
  intro h1,
  cases h with a h2,
  apply h2,
```

```
  exact h1 a,
end

-- 2ª demostración
example
  (h: ∃ x, ¬(P x))
  : ¬(∀ x, P x) :=
begin
  intro h1,
  cases h with a h2,
  exact h2 (h1 a),
end

-- 3ª demostración
example :
  (∃ x, ¬(P x)) → ¬(∀ x, P x) :=
begin
  rintro ⟨a, h2⟩ h1,
  exact h2 (h1 a),
end

-- 4ª demostración
example :
  (∃ x, ¬(P x)) → ¬(∀ x, P x) :=
λ ⟨a, h2⟩ h1, h2 (h1 a)

-- 5ª demostración
example
  (h: ∃ x, ¬(P x))
  : ¬(∀ x, P x) :=
assume h1 : ∀ x, P x,
exists.elim h
  ( assume a,
    assume h2 : ¬(P a),
    have h3 : P a,
      from h1 a,
    show false,
      from h2 h3 )

-- 6ª demostración
example
  (h: ∃ x, ¬(P x))
  : ¬(∀ x, P x) :=
assume h1 : ∀ x, P x,
exists.elim h
```

```
  ( assume a,
    assume h2 : ¬(P a),
    h2 (h1 a) )

-- 7ª demostración
example
  (h: ∃ x, ¬(P x))
  : ¬(∀ x, P x) :=
assume h1 : ∀ x, P x,
exists.elim h
  (λ a h2, h2 (h1 a) )

-- 8ª demostración
example
  (h: ∃ x, ¬(P x))
  : ¬(∀ x, P x) :=
λ h1, exists.elim h (λ a h2, h2 (h1 a) )

-- 9ª demostración
example
  (h: ∃ x, ¬(P x))
  : ¬(∀ x, P x) :=
-- by library_search
not_forall.mpr h

-- 10ª demostración
example
  (h: ∃ x, ¬(P x))
  : ¬(∀ x, P x) :=
-- by hint
by tauto

-- 11ª demostración
example
  (h: ∃ x, ¬(P x))
  : ¬(∀ x, P x) :=
by finish
```

2.2. ∀ x, P x ⊢ ∀ y, P y

```
-- Ejercicio. Demostrar
--    ∀ x, P x ⊢ ∀ y, P y
-- ----------------------------------------------------

import tactic

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

-- 1ª demostración
example
  (h : ∀ x, P x)
  : ∀ y, P y :=
begin
  intro,
  exact h y,
end
```

Enlaces al código y a la sesión en Lean Web.

2.3. $\forall$ x, P x $\to$ Q x $\vdash$ ($\forall$ x, $\neg$(Q x)) $\to$ ($\forall$ x, $\neg$(P x))

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∀ x, P x → Q x ⊢ (∀ x, ¬(Q x)) → (∀ x, ¬(P x))
-- ----------------------------------------------------

import tactic

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

-- 1ª demostración
example
  (h : ∀ x, P x → Q x)
  : (∀ x, ¬(Q x)) → (∀ x, ¬(P x)) :=
begin
  intros h1 a h2,
  apply h1 a,
  apply h a,
  exact h2,
end
```

Enlaces al código y a la sesión en Lean Web.

2.4. ∀ x, P x → ¬(Q x) ⊢ ¬(∃ x, P x ∧ Q x)

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∀ x, P x → ¬(Q x) ⊢ ¬(∃ x, P x ∧ Q x)
-- ----------------------------------------------------

import tactic

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

-- 1ª demostración
example
  (h : ∀ x, P x → ¬(Q x))
  : ¬(∃ x, P x ∧ Q x) :=
begin
  intro h1,
  cases h1 with a h2,
  apply h a,
  { exact h2.1, },
  { exact h2.2, },
end

-- 2ª demostración
example
  (h : ∀ x, P x → ¬(Q x))
  : ¬(∃ x, P x ∧ Q x) :=
begin
  rintro ⟨a,h1,h2⟩,
  apply h a,
  assumption,
  assumption,
end
```

Enlaces al código y a la sesión en Lean Web.

2.5. ∀ x y, P x y ⊢ ∀ u v, P u v

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∀ x y, P x y ⊢ ∀ u v, P u v
-- ----------------------------------------------------
```

```
import tactic

variable {U : Type}
variable {P : U → U → Prop}

-- 1ª demostración
example
  (h : ∀ x y, P x y)
  : ∀ u v, P u v :=
begin
  intros a b,
  exact h a b,
end

-- 2ª demostración
example
  (h : ∀ x y, P x y)
  : ∀ u v, P u v :=
λ a b, h a b

-- 3ª demostración
example
  (h : ∀ x y, P x y)
  : ∀ u v, P u v :=
h
```

Enlaces al código y a la sesión en Lean Web.

2.6. ∃ x y, P x y ⊢ ∃ u v, P u v

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∃ x y, P x y ⊢ ∃ u v, P u v
-- ----------------------------------------------------

import tactic

variable {U : Type}
variable {P : U → U → Prop}

-- 1ª demostración
example
  (h : ∃ x y, P x y)
```

```
  : ∃ u v, P u v :=
begin
  rcases h with ⟨a,b,h1⟩,
  use [a,b],
  exact h1,
end
```

Enlaces al código y a la sesión en Lean Web.

2.7. ∃ x, ∀ y, P x y ⊢ ∀ y, ∃ x, P x y

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∃ x, ∀ y, P x y ⊢ ∀y, ∃ x, P x y
-- ----------------------------------------------------

import tactic

variable {U : Type}
variable {P : U → U → Prop}

-- 1ª demostración
example
  (h : ∃ x, ∀ y, P x y)
  : ∀y, ∃ x, P x y :=
begin
  intro b,
  cases h with a h1,
  use a,
  exact h1 b,
end
```

Enlaces al código y a la sesión en Lean Web.

2.8. ∃x, P a → Q x ⊢ P a → (∃ x, Q x)

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∃ x, P a → Q x ⊢ P a → (∃ x, Q x)
-- ----------------------------------------------------
```

```
import tactic

variable  (U : Type)
variable  (a : U)
variables (P Q : U -> Prop)

-- 1ª demostración
example
  (h : ∃ x, P a → Q x)
  : P a → (∃ x, Q x) :=
begin
  intro h1,
  cases h with b h2,
  use b,
  apply h2,
  exact h1,
end
```

Enlaces al código y a la sesión en Lean Web.

2.9. P a → (∃ x, Q x) ⊢ ∃ x, P a → Q x

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    P a → (∃ x, Q x) ⊢ ∃ x, P a → Q x
-- ----------------------------------------------------

import tactic

variable  (U : Type)
variable  (a : U)
variables (P Q : U -> Prop)

-- 1ª demostración
example
  (h : P a → (∃ x, Q x))
  : ∃ x, P a → Q x :=
begin
  by_cases h1 : P a,
  { cases (h h1) with b h2,
    use b,
    intro,
    exact h2, },
```

```
  { use a, },
end
```

Enlaces al código y a la sesión en Lean Web.

2.10. P a → (∃ x, Q x) ⊢ ∃ x, P a → Q x

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    P a → (∃ x, Q x) ⊢ ∃ x, P a → Q x
-- ----------------------------------------------------

import tactic

variable  (U : Type)
variable  (a : U)
variables (P Q : U -> Prop)

-- 1ª demostración
example
  (h : P a → (∃ x, Q x))
  : ∃ x, P a → Q x :=
begin
  by_cases h1 : P a,
  { cases (h h1) with b h2,
    use b,
    intro,
    exact h2, },
  { use a, },
end
```

Enlaces al código y a la sesión en Lean Web.

2.11. (∃ x, P x) → Q a ⊢ ∀ x, P x → Q a

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    (∃ x, P x) → Q a ⊢ ∀ x, P x → Q a
-- ----------------------------------------------------

import tactic
```

```
variable  (U : Type)
variable  (a : U)
variables (P Q : U -> Prop)

-- 1ª demostración
example
  (h : (∃ x, P x) → Q a)
  : ∀ x, P x → Q a :=
begin
  intros b h1,
  apply h,
  use b,
  exact h1,
end
```

Enlaces al código y a la sesión en Lean Web.

2.12. ∀ x, P x → Q a ⊢ ∃ x, P x → Q a

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--     ∀ x, P x → Q a ⊢ ∃ x, P x → Q a
-- ----------------------------------------------------

import tactic

variable  (U : Type)
variable  (a : U)
variables (P Q : U -> Prop)

-- 1ª demostración
example
  (h : ∀ x, P x → Q a)
  : ∃ x, P x → Q a :=
begin
  by_cases h1: P a,
  { use a,
    exact h a, },
  { use a, },
end
```

Enlaces al código y a la sesión en Lean Web.

2.13. (∀ x, P x) ∨ (∀ x, Q x) ⊢ ∀ x, P x ∨ Q x

```
-- ---------------------------------------------------
-- Ejercicio. Demostrar
--    (∀ x, P x) ∨ (∀ x, Q x) ⊢ ∀ x, P x ∨ Q x
-- ---------------------------------------------------

import tactic

variable  (U : Type)
variables (P Q : U → Prop)

-- 1ª demostración
example
  (h : (∀ x, P x) ∨ (∀ x, Q x))
  : ∀ x, P x ∨ Q x :=
begin
  intro a,
  rcases h with (h1 | h2),
  { left,
    exact h1 a, },
  { right,
    exact h2 a, },
end
```

Enlaces al código y a la sesión en Lean Web.

2.14. ∃ x, P x ∧ Q x ⊢ (∃ x, P x) ∧ (∃ x, Q x)

```
-- ---------------------------------------------------
-- Ejercicio. 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
  (h : ∃ x, P x ∧ Q x)
  : (∃ x, P x) ∧ (∃ x, Q x) :=
```

```
begin
  rcases h with ⟨a, hP, hQ⟩,
  split,
  { use a,
    exact hP, },
  { use a,
    exact hQ, },
end

-- 1ª demostración
example
  (h : ∃ x, P x ∧ Q x)
  : (∃ x, P x) ∧ (∃ x, Q x) :=
begin
  rcases h with ⟨a, hP, hQ⟩,
  exact ⟨⟨a, hP⟩, ⟨a,hQ⟩⟩,
end
```

Enlaces al código y a la sesión en Lean Web.

2.15. ∀ x y, P y → Q x ⊢ (∃ y, P y) → (∀ x, Q x)

```
-- ---------------------------------------------------
-- Ejercicio. Demostrar
--    ∀ x y, P y → Q x ⊢ (∃ y, P y) → (∀ x, Q x)
-- ---------------------------------------------------

import tactic

variable  (U : Type)
variables (P Q : U → Prop)

-- 1ª demostración
example
  (h : ∀ x y, P y → Q x)
  : (∃ y, P y) → (∀ x, Q x) :=
begin
  intros h1 a,
  cases h1 with b h2,
  apply (h a b),
  exact h2,
end
```

```
-- 2ª demostración
example
  (h : ∀ x y, P y → Q x)
  : (∃ y, P y) → (∀ x, Q x) :=
begin
  rintro ⟨b, h1⟩ a,
  exact (h a b) h1,
end

-- 3ª demostración
example
  (h : ∀ x y, P y → Q x)
  : (∃ y, P y) → (∀ x, Q x) :=
λ ⟨b, h1⟩ a, (h a b) h1
```

Enlaces al código y a la sesión en Lean Web.

2.16. ¬(∀ x, ¬(P x)) ⊢ ∃ x, P x

```
-- ---------------------------------------------------
-- Ejercicio. Demostrar
--     ¬(∀ x, ¬(P x)) ⊢ ∃ x, P x
-- ---------------------------------------------------

import tactic

variable (U : Type)
variable (P : U -> Prop)

open_locale classical

-- 1ª demostración
example
  (h: ¬(∀ x, ¬(P x)))
  : ∃ x, P x :=
begin
  by_contradiction h1,
  apply h,
  intros a h2,
  apply h1,
  use a,
  exact h2,
end
```

```
-- 2ª demostración
example
  (h: ¬(∀ x, ¬(P x)))
  : ∃ x, P x :=
begin
  push_neg at h,
  exact h,
end

-- 3ª demostración
example
  (h: ¬(∀ x, ¬(P x)))
  : ∃ x, P x :=
-- by library_search
not_forall_not.mp h
```

Enlaces al código y a la sesión en Lean Web.

2.17. ∀ x, ¬(P x) ⊢ ¬(∃ x, P x)

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∀ x, ¬(P x) ⊢ ¬(∃ x, P x)
-- ----------------------------------------------------

import tactic

variable (U : Type)
variable (P : U -> Prop)

-- 1ª demostración
example
  (h : ∀ x, ¬(P x))
  : ¬(∃ x, P x) :=
begin
  rintro ⟨a, h1⟩,
  apply h a,
  exact h1,
end

-- 2ª demostración
example
```

```
  (h : ∀ x, ¬(P x))
  : ¬(∃ x, P x) :=
begin
  push_neg,
  exact h,
end

-- 3ª demostración
example
  (h : ∀ x, ¬(P x))
  : ¬(∃ x, P x) :=
-- by library_search
not_exists.mpr h
```

Enlaces al código y a la sesión en Lean Web.

2.18. ∃ x, P x ⊢ ¬(∀ x, ¬(P x))

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∃ x, P x ⊢ ¬(∀ x, ¬(P x))
-- ----------------------------------------------------

import tactic

variable (U : Type)
variable (P : U -> Prop)

-- 1ª demostración
example
  (h : ∃ x, P x)
  : ¬(∀ x, ¬(P x)) :=
begin
  intro h1,
  cases h with a h2,
  apply h1 a,
  exact h2,
end
```

Enlaces al código y a la sesión en Lean Web.

2.19. P a → (∀ x, Q x) ⊢ ∀ x, P a → Q x

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    P a → (∀ x, Q x) ⊢ ∀ x, P a → Q x
-- ----------------------------------------------------

import tactic

variable  (U : Type)
variable  (a : U)
variables (P Q : U -> Prop)

-- 1ª demostración
example
  (h : P a → (∀ x, Q x))
  : ∀ x, P a → Q x :=
begin
  rintro b h1,
  exact (h h1) b,
end
```

Enlaces al código y a la sesión en Lean Web.

2.20. ∀ x y z, R x y ∧ R y z → R x z; ∀ x, ¬(R x x) ⊢ ∀ x y, R x y → ¬(R y x)

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--      {∀ x y z, R x y ∧ R y z → R x z,
--       ∀ x, ¬(R x x)}
--      ⊢ ∀ x y, R x y → ¬(R y x)
-- ----------------------------------------------------

import tactic

variable {U : Type}
variable {R : U → U → Prop}

-- 1ª demostración
example
  (h1 : ∀ x y z, R x y ∧ R y z → R x z)
```

```
  (h2 : ∀ x, ¬(R x x))
  : ∀ x y, R x y → ¬(R y x) :=
begin
  intros a b h3 h4,
  apply h2 a,
  apply h1 a b a,
  exact ⟨h3, h4⟩,
end
```

Enlaces al código y a la sesión en Lean Web.

2.21. ∀x, P x ∨ Q x; ∃x, ¬Q x, ∀ x, R x → ¬P x ⊢ ∃x, ¬R x

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∀x, P x ∨ Q x; ∃x, ¬Q x; ∀ x, R x → ¬P x ⊢ ∃x, ¬R x
-- ----------------------------------------------------

import tactic

variable  (U : Type)
variables (P Q R : U → Prop)

-- 1ª demostración
example
  (h1 : ∀x, P x ∨ Q x)
  (h2 : ∃x, ¬Q x)
  (h3 : ∀ x, R x → ¬P x)
  : ∃x, ¬R x  :=
begin
  cases h2 with a h4,
  use a,
  intro h5,
  apply h4,
  cases (h1 a) with h6 h7,
  { exfalso,
    apply (h3 a) h5,
    exact h6, },
  { exact h7, },
end
```

Enlaces al código y a la sesión en Lean Web.

2.22. ∀x, P x → Q x ∨ R x; ¬∃x, P x ∧ R x ⊢ ∀x, P x → Q x

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∀x, P x → Q x ∨ R x; ¬∃x, P x ∧ R x ⊢ ∀x, P x → Q x
-- ----------------------------------------------------

import tactic

variable  (U : Type)
variables (P Q R : U → Prop)

-- 1ª demostración
example
  (h1 : ∀x, P x → Q x ∨ R x)
  (h2 : ¬∃x, P x ∧ R x)
  : ∀x, P x → Q x :=
begin
  intros a h3,
  cases (h1 a) h3 with h4 h5,
  { exact h4, },
  { exfalso,
    apply h2,
    use a,
    exact ⟨h3, h5⟩, },
end
```

Enlaces al código y a la sesión en Lean Web.

2.23. ∃ x y, R x y ∨ R y x ⊢ ∃ x y, R x y

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ∃ x y, R x y ∨ R y x ⊢ ∃ x y, R x y
-- ----------------------------------------------------

import tactic

variable {U : Type}
variable {R : U → U → Prop}
```

```
-- 1ª demostración
example
  (h : ∃ x y, R x y ∨ R y x)
  : ∃ x y, R x y :=
begin
  rcases h with ⟨a, b, (h1 | h2)⟩,
  { use [a, b],
    exact h1, },
  { use [b, a],
    exact h2, },
end
```

Enlaces al código y a la sesión en Lean Web.

2.24. (∃x, ∀y, P x y) → (∀y, ∃x, P x y)

```
-- ---------------------------------------------------
-- Ejercicio. Demostrar
--    (∃x, ∀y, P x y) → (∀y, ∃x, P x y)
-- ---------------------------------------------------

import tactic

variable {U : Type}
variable {P : U → U → Prop}

-- 1ª demostración
example :
  (∃x, ∀y, P x y) → (∀y, ∃x, P x y) :=
begin
  intros h b,
  cases h with a h1,
  use a,
  exact h1 b,
end
```

Enlaces al código y a la sesión en Lean Web.

2.25. $(\forall$ x, P x $\to$ Q) $\longleftrightarrow$ (($\exists$x, P x) $\to$ Q)

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    (∀ x, P x → Q) ↔ ((∃x, P x) → Q)
-- ----------------------------------------------------

import tactic

variable (U : Type)
variable (P : U -> Prop)
variable (Q : Prop)

-- 1ª demostración
example :
  (∀ x, P x → Q) ↔ ((∃x, P x) → Q) :=
begin
  split,
  { rintros h1 ⟨a, h2⟩,
    exact h1 a h2, },
  { intros h3 a h4,
    apply h3,
    use a,
    exact h4, },
end
```

Enlaces al código y a la sesión en Lean Web.

2.26. (($\forall$x, P x) $\wedge$ ($\forall$x, Q x)) $\longleftrightarrow$ ($\forall$x, P x $\wedge$ Q x)

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar
--    ((∀x, P x) ∧ (∀x, Q x)) ↔ (∀x, P x ∧ Q x)
-- ----------------------------------------------------

import tactic

variable  (U : Type)
variables (P Q : U → Prop)

-- 1ª demostración
example :
  ((∀x, P x) ∧ (∀x, Q x)) ↔ (∀x, P x ∧ Q x) :=
```

```
begin
  split,
  { rintros ⟨h1, h2⟩ a,
    exact ⟨h1 a, h2 a⟩, },
  { intro h3,
    split,
    { intro a,
      exact (h3 a).left, },
    { intro a,
      exact (h3 a).right, }},
end
```

Enlaces al código y a la sesión en Lean Web.

2.27. ((∃x, P x) ∨ (∃x, Q x)) ⟷ (∃x, P x ∨ Q x)

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar o refutar
--    ((∃x, P x) ∨ (∃x, Q x)) ↔ (∃x, P x ∨ Q x)
-- ----------------------------------------------------

import tactic

variable  (U : Type)
variables (P Q : U → Prop)

-- 1ª demostración
example :
  ((∃x, P x) ∨ (∃x, Q x)) ↔ (∃x, P x ∨ Q x) :=
begin
  split,
  { rintro (⟨a, h1⟩ | ⟨a, h2⟩),
    { use a,
      left,
      exact h1, },
    { use a,
      right,
      exact h2, }},
  { rintro ⟨a, (h3 | h4)⟩,
    { left,
      use a,
      exact h3, },
    { right,
```

```
      use a,
      exact h4, }},
end
```

Enlaces al código y a la sesión en Lean Web.

2.28. (¬(∀ x, P x)) ⟷ (∃x, ¬P x)

```
-- ----------------------------------------------------
-- Ejercicio 30. Demostrar o refutar
--      (¬(∀ x, P x)) ↔ (∃x, ¬P x)
-- ----------------------------------------------------

import tactic

variable (U : Type)
variable (P : U -> Prop)

open_locale classical

-- 1ª demostración
example :
  (¬(∀ x, P x)) ↔ (∃x, ¬P x) :=
begin
  split,
  { intro h1,
    by_contradiction h2,
    apply h1,
    intro a,
    by_contradiction h3,
    apply h2,
    use a, },
  { rintro ⟨a, h4⟩ h5,
    apply h4,
    exact h5 a, },
end

-- 2ª demostración
example :
  (¬(∀ x, P x)) ↔ (∃x, ¬P x) :=
begin
  split,
  { intro h1,
```

```
    push_neg at h1,
    exact h1, },
  { intro h2,
    push_neg,
    exact h2, },
end

-- 3ª demostración
example :
  (¬(∀ x, P x)) ↔ (∃x, ¬P x) :=
begin
  push_neg,
  trivial,
end

-- 4ª demostración
example :
  (¬(∀ x, P x)) ↔ (∃x, ¬P x) :=
-- by library_search
not_forall
```

Enlaces al código y a la sesión en Lean Web.

Capítulo 3

Ejercicios sobre igualdad y funciones

3.1. P a ⊢ ∀ x, x = a → P x

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar o refutar
--    P a ⊢ ∀ x, x = a → P x
-- ----------------------------------------------------

import tactic

variable  (U : Type)
variable  (a : U)
variables (P Q : U -> Prop)

-- 1ª demostración
example
  (h : P a)
  : ∀ x, x = a → P x :=
begin
  intros b h1,
  rw h1,
  exact h,
end

-- 1ª demostración
example
  (h : P a)
  : ∀ x, x = a → P x :=
begin
```

```
  intros b h1,
  rwa h1,
end
```

Enlaces al código y a la sesión en Lean Web.

3.2. ∃x y, R x y ∨ R y x; ¬∃x, R x x ⊢ ∃x y, x ≠ y

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar o refutar
--    ∃x y, R x y ∨ R y x; ¬∃x, R x x ⊢ ∃x y, x ≠ y
-- ----------------------------------------------------

import tactic

variable {U : Type}
variable {R : U → U → Prop}

-- 1ª demostración
example
  (h1 : ∃x y, R x y ∨ R y x)
  (h2 : ¬(∃x, R x x))
  : ∃ (x : U) y, (x ≠ y) :=
begin
  rcases h1 with ⟨a, b, h3⟩,
  use [a, b],
  intro h4,
  apply h2,
  use b,
  cases h3 with h5 h6,
  { rwa h4 at h5, },
  { rwa h4 at h6, },
end
```

Enlaces al código y a la sesión en Lean Web.

3.3. ∀x, P a x x; ∀xyz, P x y z → P (f x) y (f z) ⊢ P (f a) a (f a)

```
-- ---------------------------------------------------
-- Ejercicio. Demostrar o refutar
--     ∀x, P a x x;
--     ∀xyz, P x y z → P (f x) y (f z)
--     ⊢ P (f a) a (f a)
-- ---------------------------------------------------

import tactic

variable (U : Type)
variable (P : U → U → U → Prop)
variable (a : U)
variable (f : U → U)

-- 1ª demostración
example
  (h1 : ∀ x, P a x x)
  (h2 : ∀ x y z, P x y z → P (f x) y (f z))
  : P (f a) a (f a) :=
begin
  apply h2,
  exact h1 a,
end

-- 2ª demostración
example
  (h1 : ∀ x, P a x x)
  (h2 : ∀ x y z, P x y z → P (f x) y (f z))
  : P (f a) a (f a) :=
h2 a a a (h1 a)
```

Enlaces al código y a la sesión en Lean Web.

3.4. ∀x, P a x x; ∀xyz, P x y z → P (f x) y (f z) ⊢ ∃z, P (f a) z (f (f a))

```
-- ---------------------------------------------------
-- Ejercicio. Demostrar o refutar
--     ∀x, P a x x; ∀xyz, P x y z → P (f x) y (f z) ⊢ ∃z, P (f a) z (f (f a))
-- ---------------------------------------------------

import tactic
```

```
variable (U : Type)
variable (P : U → U → U → Prop)
variable (a : U)
variable (f : U → U)

-- 1ª demostración
example
  (h1 : ∀ x, P a x x)
  (h2 : ∀ x y z, P x y z → P (f x) y (f z))
  : ∃ z, P (f a) z (f (f a)) :=
begin
  use f a,
  apply h2,
  exact h1 (f a),
end
```

Enlaces al código y a la sesión en Lean Web.

3.5. ∀y, Q a y; ∀xy, Q x y → Q (s x) (s y) ⊢ ∃z, Qa z ∧ Q z (s (s a))

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar o refutar
--    ∀y, Q a y;
--    ∀xy, Q x y → Q (s x) (s y)
--    ⊢ ∃z, Q a z ∧ Q z (s (s a))
-- ----------------------------------------------------

import tactic

variable (U : Type)
variable (Q : U → U → Prop)
variable (a : U)
variable (s : U → U)

-- 1ª demostración
example
  (h1 : ∀ y, Q a y)
  (h2 : ∀ x y, Q x y → Q (s x) (s y))
  : ∃z, Q a z ∧ Q z (s (s a)) :=
begin
  use a,
```

```
  split,
  { exact h1 a, },
  { exact h1 (s (s a)), },
end
```

Enlaces al código y a la sesión en Lean Web.

3.6. x = f x; P (f x) ⊢ P x

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar o refutar
--    x = f x; P (f x) ⊢ P x
-- ----------------------------------------------------

import tactic

variable (U : Type)
variable (P :  U → Prop)
variable (f : U → U)
variable (x : U)

-- 1ª demostración
example
  (h1 : x = f x)
  (h2 : P (f x))
  : P x :=
begin
  rw h1,
  exact h2,
end

-- 2ª demostración
example
  (h1 : x = f x)
  (h2 : P (f x))
  : P x :=
begin
  rwa h1,
end

-- 3ª demostración
example
  (h1 : x = f x)
```

```
  (h2 : P (f x))
  : P x :=
by rwa h1
```

Enlaces al código y a la sesión en Lean Web.

3.7. x = f x, triple (f x) (f x) x ⊢ triple x x x

```
-- ----------------------------------------------------
-- Ejercicio. Demostrar o refutar
--    x = f x, triple (f x) (f x) x ⊢ triple x x x
-- ----------------------------------------------------

variable (U : Type)
variable (x : U)
variable (triple : U → U → U → Prop)
variable (f : U → U)

-- 1ª demostración
example
  (h1 : x = f x)
  (h2 : triple (f x) (f x) x)
  : triple x x x :=
begin
  rw ← h1 at h2,
  exact h2,
end

-- 2ª demostración
example
  (h1 : x = f x)
  (h2 : triple (f x) (f x) x)
  : triple x x x :=
begin
  rwa ← h1 at h2,
end

-- 3ª demostración
example
  (h1 : x = f x)
  (h2 : triple (f x) (f x) x)
  : triple x x x :=
by rwa ← h1 at h2
```

Enlaces al código y a la sesión en Lean Web.

Capítulo 4

Bibliografía

- Deducción natural en lógica de primer orden. ~ J.A. Alonso, A. Cordón, M.J. Hidalgo.
- Lógica con Lean ~ J.A. Alonso.
  - Cap. 2: Lógica proposicional.
- Logic and proof. ~ J. Avigad, R.Y. Lewis, F. van Doorn.
  - Cap. 4: Propositional Logic in Lean.
- Logic in Computer Science. ~ M. Huth, M. Ryan.
  - Cap. 1.2: Propositional logic. Natural deduction.
- Theorem proving in Lean. ~ J. Avigad, L. de Moura, S. Kong.
  - Cap. 3: Propositions and proofs.