Lógica con Lean

José A. Alonso Jiménez

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, 13 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. 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
```

2.2. 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
```