

# 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

Esta obra está bajo una licencia Reconocimiento–NoComercial–CompartirIgual 2.5 Spain de Creative Commons.

**Se permite:**

- copiar, distribuir y comunicar públicamente la obra
- hacer obras derivadas

**Bajo las condiciones siguientes:**

**Reconocimiento.** Debe reconocer los créditos de la obra de la manera especificada por el autor.



**No comercial.** No puede utilizar esta obra para fines comerciales.



**Compartir bajo la misma licencia.** Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

- Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.
- Algunas de estas condiciones pueden no aplicarse si se obtiene el permiso del titular de los derechos de autor.

Esto es un resumen del texto legal (la licencia completa). Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-nc-sa/2.5/es/> o envíe una carta a Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

# Índice general

<b>1</b>	<b>Introducción</b>	<b>5</b>
<b>2</b>	<b>Lógica proposicional</b>	<b>7</b>
2.1	Reglas del condicional . . . . .	7
2.1.1	Regla de eliminación del condicional . . . . .	7
2.1.2	Regla de introducción del condicional . . . . .	9
2.2	Reglas de la conjunción . . . . .	10



# 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 prerequisite 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 ejercicio 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

- Enlaces al [código](#), a la [sesión en Lean Web](#) y al [vídeo](#).

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

-- -----

-- Ejercicio 1. Demostrar que
--    $(P \rightarrow Q), P \vdash Q$ .
-- -----

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

- Enlaces al [código](#), a la [sesión en Lean Web](#) y al [vídeo](#).

```
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

- Enlaces al [código](#), a la [sesión en Lean Web](#) y al [vídeo](#).

```

-- Reglas de la conjunción
-- =====

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