Trabajo de Fin de Grado


# Formalización de las matemáticas con Lean. Un caso de estudio: Geometría euclídea plana.

Adrián Lattes Grassi

Septiembre de 2023

Facultad de Ciencias Matemáticas.
Trabajo dirigido por Jorge Carmona Ruber.

**Resumen**

Este trabajo explora el uso del asistente de demostración *Lean*, un lenguaje de programación que implementa una teoría de tipos útil para verificar formalmente demostraciones matemáticas, para formalizar enunciados y resultados de la axiomática de Hilbert de la geometría euclídea plana. Esta teoría servirá de guía para introducir el uso del asistente y exponer cómo puede ser utilizado para construir relaciones de equivalencia, modelos de una teoría y demostrar la independencia entre axiomas.

**Abstract**

Resumen traducido al inglés.

# Índice

# Metodología y plan de trabajo

# 1. Matemáticas y geometría formales.

La formalización matemática es el proceso de representar enunciados y demostraciones matemáticas utilizando un lenguaje formal y un conjunto de reglas lógicas bien definidas. En este contexto, se establece un conjunto de proposiciones fundamentales, conocidas como axiomas, que se aceptan sin requerir justificación. Estos axiomas se manipulan y transforman mediante la aplicación sistemática de las reglas lógicas para derivar nuevas proposiciones matemáticas. El proceso de aplicar secuencialmente estas reglas lógicas para obtener una conclusión a partir de proposiciones más básicas es conocido como demostración.

En la historia de las matemáticas los *Elementos* de *Euclides*, tratado matemático compuesto de trece libros escrito en el siglo III a.C., son el ejemplo más antiguo de proyecto de formalización matemática. En la obra se trata rigurosamente una extensa variedad de temas, como la geometría plana y espacial o la teoría de números. Euclides es el primer autor en presentar los conocimientos matemáticos siguiendo un método formal. En los tratados se presentan los argumentos a partir de una serie de postulados, definiciones y nociones comunes a partir de los cuales se demuestran proposiciones y teoremas mediante razonamientos deductivos.

La obra de Euclides ha tenido una profunda influencia en las matemáticas, la lógica y la filosofía. Los *Elementos* se mantuvieron como la principal referencia en geometría durante casi dos milenios. Las pruebas rigurosas, apoyadas en el razonamiento lógico y la estructura del tratado establecieron un estándar para la argumentación y la exposición matemática que todavía se conserva en la actualidad.

A lo largo de la historia, se ha evidenciado que los *Elementos* de *Euclides* no se ciñen estrictamente al método axiomático. En el tratado se encuentran razonamientos sustentados en intuiciones geométricas y construcciones con regla y compás, en lugar de en deducciones estrictamente lógicas derivadas de los postulados.

Durante siglos, matemáticos y filósofos han examinado y escrutado la obra de Euclides, identificando errores y omisiones y planteando cuestiones sobre la relación entre los postulados. El quinto postulado de Euclides, también conocido como postulado de las paralelas, ha sido sometido a un análisis riguroso. Este postulado afirma que, dada una línea recta y un punto fuera de ella, existe únicamente una línea recta paralela a la línea dada que pasa por el punto en cuestión. Por siglos, numerosos matemáticos, intuyendo la innecesariedad del postulado, intentaron demostrarlo a partir de los otros cuatro, pero sin éxito. El problema fue resuelto en el siglo XIX con la concepción de nuevas geometrías, como la hiperbólica y la elíptica, en las cuales el postulado de las paralelas no se verifica. Quedó así demostrada la independencia del quinto postulado

respecto a los primeros cuatro, y por tanto su necesidad para la construcción de la geometría euclídea.

Durante el siglo XIX se dieron avances trascendentales en el desarrollo de las matemáticas y la lógica formales. Ejemplos notables son la formulación del álgebra de Boole, la lógica de predicados propuesta por *Gottlob Frege* o el desarrollo de la *aritmética de Peano*. En este contexto el matemático alemán *David Hilbert* publicó su obra *Grundlagen der Geometrie* (Fundamentos de Geometría), en la cual se lleva a cabo una revisión exhaustiva de los *Elementos*, planteando una nueva axiomatización para formalizar correctamente los resultados de la geometría euclídea, eliminando por completo el recurso a la intuición y razonamientos geométricos en la presentación de los resultados.

**Explicar que este es el punto de partida en el que enmarcar este trabajo. A partir de aqui se va a explicar qué aporta la formalización matemática asistida por computadores y en que sentido es un paso más.**

# 2. Formalización asistida por computadores

Los asistentes de demostración nos permiten formalizar definiciones, enunciados de proposiciones y teoremas, demostraciones, y verficar estas definiciones. Formalizar matemáticas consiste en digitalizar enunciados y resultados escribiéndolos en un lenguaje de programación que garantiza, mediante una correspondencia entre una teoríá de tipos y la lógica, la validez de cada paso.

Algunos beneficios de formalizar enunciados y resultados matemáticos mediante un asistente de demostración son:

Introducción a la formalización en Lean. Demostración asistida por computadores. ¿Pa que?

- Comprobación mecanizada de demostraciones.
- Digitalizar resultados y crear base de datos. FormalAbstracts
- Investigación en técnicas de demostración automática

# 3. Introducción a Lean

## 3.1. Teoría de tipos informal

Lo primero que se necesita en matemáticas para poder formalizar afirmaciones es un lenguaje formal con el cual expresarlas. Normalmente se utiliza la lógica de primer orden con los axiomas de la teoría de conjuntos. Lean, sin embargo, utiliza un sistema deductivo diferente, el de la teoría de tipos. **Pendiente:** especificar siempre qué teoría a que teoría de tipos me refiero

En lógica de primer orden se tiene una aserción fundamental, *que una proposición dada tenga una prueba*. Es decir, cada proposición $P$ da lugar a la aserción correspondiente *$P$ tiene una prueba*. Mediante ciertas reglas de transformación, y a veces una serie de axiomas, se pueden construir nuevas pruebas. Por ejemplo, dada la regla de inferencia *de $A$ se deduce $A$ o $B$* y la aserción *$A$ tiene una prueba*, se obtiene la aserción *$A$ o $B$ tiene una prueba*.

En teoría de tipos la aserción fundamental es *que un término tenga un tipo*. Dados un término $a$ y un tipo $A$, si *$a$ tiene tipo $A$* escribimos $a : A$. Esta es misma notación es la utilizada por Lean, en el que por ejemplo podemos expresar la aserción *3 es un número natural* con el código `3 : ℕ`.

Es importante no confundir esta notación con la de una relación interna a nuestro lenguaje. Mientras que en teoría de conjuntos utilizamos la relación de pertenencia $\in$ para expresar que un elemento primitivo (un conjunto) está contenido en otro, en teoría de tipos no podemos considerar los términos o los tipos por separado. La noción fundamental es la pertenencia de tipos y cada término tiene que estar siempre acompañado por su tipo. En teoría de tipos además existen otras aserciones, como la de igualdad entre términos de un mismo tipo. Dados $a : A$ y $b : A$, se tiene la aserción *$a$ y $b$ son dos términos de tipo $A$ iguales por definición*, y escribimos $a \equiv b : A$.

**Pendiente:** Mencionar que la igualdad es un tema delicado.

La teoría de tipos también puede utilizarse para expresar afirmaciones y demostraciones matemáticas. Las afirmaciones se codifican mediante los tipos y las demostraciones mediante construcciones de términos de un tipo dado. Es decir, se puede interpretar la aserción $a : A$ como *$a$ es una demostración de $A$*. Esta interpretación da lugar a una analogía entre la lógica proposicional y la teoría de tipos, llamada correspondencia de Curry-Howard. Sin entrar en detalles, a cada proposición lógica se le puede asignar un tipo, y a cada demostración de un enunciado un término del tipo correspondiente al enunciado.

**Referencia:** HTT

Existen distintas elecciones de reglas de transformación que considerar en teoría de tipos, que dan lugares a distintas versiones de la teoría de tipos. Lean implementa una una versión de la teoría de tipos dependiente conocida como el *Calculus of Constructions*.

La base de muchas teorías de tipos es el lambda cálculo, un modelo universal de computación introducido por *Alonzo Church* en los años treinta. Sin entrar en su formalización, en el lambda cálculo se consideran dos operaciones fundamentales para tratar con funciones, la abstracción y la evaluación.

- **Abstracción**. Es el mecanismo de definición de funciones mediante la introducción de parámetros. Dado un término `x + 1 : ℕ`, mediante la sintaxis `λ x : ℕ, (x + 1 : ℕ)` se convierte la variable libre `x` en una variable ligada por la abstracción, a la que llamamos parámetro de la función.

  Es importante recordar que cada término tiene que ir acompañado del tipo al que pertenece. En esta expresión estamos indicando que tanto el parámetro `x` como el resultado de la función, `x+1`, son de tipo `ℕ`. Es decir, tenemos una función que dado un número natural devuelve otro número natural. Esto también puede escribirse como `(λ x, x + 1) : ℕ → ℕ`.

  Lean además incluye notación para definir funciones que devuelven otras funciones, lo cual es muy útil para tratar funciones que reciben más de un parámetro (**Referencia:** Ver currificacion). Las siguientes líneas de código definen expresiones equivalentes

```
λ a : α, λ b : β, a
λ (a : α) (b : β), a
```

  que representan el mismo término, de tipo $\alpha \to \beta \to \alpha$.

- **Evaluación**. Consiste en aplicar funciones, pasándoles los valores de los argumentos que evaluar. Por ejemplo la expresión `(λ x : ℕ, (x + 1) : ℕ) (1 : ℕ)` indica que estamos evaluando la función `(λ x, x + 1) : ℕ → ℕ` con el parámetro `x` sustituido por el término `1 : ℕ` (para que la sustitución pueda realizarse los tipos deben coincidir). Mediante un proceso de reducción se obtiene el término `2 : ℕ`.

**Referencia:** Theorem proving in Lean [AvD]

Como se ve en los ejemplos, el código fuente de Lean se pueden incluir caracteres unicode, como λ, → o ℕ. Esta característica del lenguaje es muy útil a la hora de escribir código lo más cercano posible a las notaciones a las que estamos acostumbrados a usar en matemáticas. La inclusión de estos caracteres está facilitada en el entorno de desarrollo, escribiendo comandos que empiecen por

\ estos se reemplazarán por el caracter correspondiente. Por ejemplo al escribir `\to` este comando se reemplazará automáticamente por el caracter →, `\lambda` por λ y `\N` por ℕ.

**Pendiente:** Incluir en algún lugar explicación sobre entorno de desarrollo (plugin vscode) y sus características (¿Apéndice?).

Que cada término sea siempre considerado junto a su tipo no significa que sea siempre necesario explicitar dicho tipo. Lean tiene un mecanismo llamado *inferencia de tipos* que le permite deducir automáticamente el tipo de un término cuando no ha sido explicitado pero el contexto aporta información suficiente. Por ejemplo, cuando definimos la función `λ x : ℕ, (x + 1 : ℕ)` no es necesario incluir la segunda anotación de tipo. Dada la expresión `x + 1` y sabiendo por contexto que `x : ℕ` el sistema de inferencia de tipos deduce que la suma de dos números naturales también es un número natural, por lo que se infiere el tipo ℕ.

## 3.2. Introducción de términos

En lean existen distintas formas de introducir nuevos términos en el entorno actual.

### Constantes

Mediante los comandos `constant` y `constants` se pueden introducir términos en el entorno, postulando su existencia. Este comando equivale por tanto a considerar nuevos axiomas sobre la existencia de los términos que introduce.

```
constant a : ℕ
constants (b : ℤ) (c : ℂ)
```

### Definiciones

Si no queremos introducir nuevos axiomas podemos definir nuevos términos mediante el comando `def`. Como estamos definiendo un símbolo, es necesario proporcionar el tipo y término que queremos asignar al símbolo:

```
def succ : ℕ → ℕ := λ n, n + 1
def succ' (n : ℕ) : ℕ := n + 1 -- Otra forma de definir succ
```

Para introducir parámetros de funciones se puede omitir la notación λ, incluyendo las variables parametrizadas antes de los dos puntos que anotan el tipo de la definición.

Esta notación de introducción de parámetros es muy útil y simple, pero puede resultar demasiado explícita. Veamos por ejemplo cómo se puede definir la función identidad, que dado un término de un tipo devuelve el mismo término. Si queremos que la identidad definida sea sea general y se le puedan aplicar términos de cualquier tipo necesitaremos introducir el tipo del término como un parámetro adicional.

```
def id1 (α : Type*) (e : α) := e
```

El problema de esta definición es que cada vez que se quiera utilizar será necesario proporcionarle como primer argumento el tipo del término que se le quiere pasar, por ejemplo `id1 ℕ 0`. Pero la función identidad que queremos considerar recibe un solo argumento. Como a cada término acompaña siempre su tipo, dado el término `e : α`, el sistema de inferencia de tipos de Lean es capaz de deducir automáticamente el tipo `α`. Solo falta indicar en la definición cuál es el parámetro cuya identificación queremos delegar al sistema de inferencia de tipos.

```
def id2 {α : Type*} (e : α) := e
```

Así los parámetros indicados entre llaves, llamados *parámetros implícitos*, serán deducidos automaticamente.

**Pendiente:** Explicar comando `variable`

## 3.3. Proposiciones

En Lean se tiene el tipo `Prop` para expresar las proposiciones mediante la analogía de *proposiciones como tipos* dada por la correspondencia de Curry-Howard.

La correspondencia de Curry-Howard afirma que estas funciones de la teoría de tipos se comportan de la misma forma que la implicación en lógica. Por tanto en Lean utilizaremos el símbolo $\to$ para referirnos tanto a funciones como a implicaciones dentro de una proposición. Dadas dos proposiciones `p q : Prop` podemos construir la proposición `p → q : Prop`, que se interpreta como `p` *implica* `q`.

**Pendiente:** Explicar cómo se expresan las proposiciones. or, and, neg, forall, exists, neq, etc

### 3.4. Demostraciones

**Pendiente:** Explicar qué es una demostración en teoría de tipos (term mode). Explicar qué son las tácticas y cómo funciona el modo táctico.

**Pendiente:** Incluir anexo hablando sobre el entorno de desarrollo y el modo táctico, con captura de pantalla.

### 3.5. Mathlib

**Pendiente:** Contar alguna cosa básica. Explicar cómo utilizar la web para encontrar cosas.

## 4. Formalizando la geometría de Hilbert en Lean

En esta sección se presentan algunos axiomas y resultados elementales de la axiomática de Hilbert, comparando los enunciados y demostraciones expresados de forma natural con sus correspondientes formalizaciones en Lean.

En 1899 Hilbert empezó a desarrollar su propuesta de nueva fundamentación de la geometría euclídea, mediante una serie de apuntes de conferencias que más tarde se convertirían en el tratado *Grundlagen der Geometrie* (Fundamentos de Geometría). Este trabajo hace énfasis en los problemas de clasificación de nociones primitivas, grupos de axiomas, interdependencias entre las distintas partes de la teoría y minimalidad de los axiomas considerados.

Esta nueva teoría geométrica parte de postular ciertas nociones primitivas y una serie de axiomas que establecen cómo se relacionan estas nociones. En este trabajo se ha seguido una versión modernizada de los axiomas y los resultadas, basada en las presentaciones de los libros de *Hartshorne* [Har00] y *Greenberg* [Gre93], en las que se considera el caso restringido de la geometría plana.

Consideraremos por tanto cinco nociones primitivas: dos términos primitivos (*puntos* y *líneas*) y cuatro relaciones primitivas (*incidencia*, *orden*, *congruencia de segmentos* y *congruencia de ángulos*).

Se tratarán los axiomas y definiciones correspondientes a estas nociones primitivas, siguiendo la estructura del tratado, mencionando alguna cuestión sobre el axioma de las paralelas, pero sin entrar en cuestiones de continuidad. Además se incluirá alguna formalización de resultados demostrables con las nociones presentadas.

# 5. Conclusiones

**Pendiente:** Redactar

# Referencias


[AvD] Jeremy Avigad y Floris van Doorn. «The Lean Theorem Prover and Homotopy Type Theory». En: ().

[Gre93] Marvin J. Greenberg. *Euclidean and Non-Euclidean Geometries: Development and History*. 3rd ed. New York: W.H. Freeman, 1993. 483 págs. ISBN: 978-0-7167-2446-9.

[Har00] Robin Hartshorne. *Geometry: Euclid and Beyond*. Red. de S. Axler, F. W. Gehring y K. A. Ribet. Undergraduate Texts in Mathematics. New York, NY: Springer New York, 2000. ISBN: 978-1-4419-3145-0 978-0-387-22676-7. DOI: 10.1007/978-0-387-22676-7. URL: http://link.springer.com/10.1007/978-0-387-22676-7 (visitado 27-03-2023).


referencias