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

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

Adrián Lattes Grassi

18 de septiembre de 2023

Formalización asistida por computadores

- Digitalización de definiciones y enunciados
- Comprobación mecanizada de demostraciones
- Uso en docencia y comunicación
- Demostración automatizada

Formalizando matemáticas en Lean

- Lean implementa el *Cálculo de construcciones inductivas*

Formalizando matemáticas en Lean

- Lean implementa el *Cálculo de construcciones inductivas*
- Correspondencia de Curry-Howard

Formalizando matemáticas en Lean

- Lean implementa el *Cálculo de construcciones inductivas*
- Correspondencia de Curry-Howard
- *Proposiciones como tipos*

```
P : Prop
```

Formalizando matemáticas en Lean

- Lean implementa el *Cálculo de construcciones inductivas*
- Correspondencia de Curry-Howard
- *Proposiciones como tipos*

```
P : Prop
```

- *Demostraciones como términos*

```
p : P : Prop
```

Formalizando matemáticas en Lean

- Lean implementa el *Cálculo de construcciones inductivas*
- Correspondencia de Curry-Howard
- *Proposiciones como tipos*

  ```
  P : Prop
  ```
- *Demostraciones como términos*

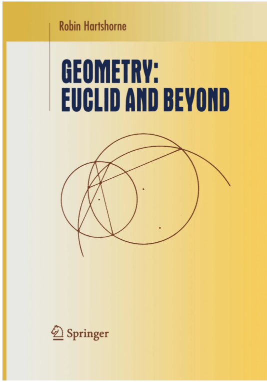
  ```
  p : P : Prop
  ```
- Modo táctico

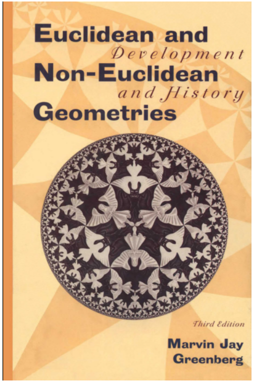
Geometría euclídea plana

Axiomatización de Hilbert

Geometría euclídea plana

Axiomatización de Hilbert





Geometría euclídea plana

Nociones fundamentales

- *Puntos*. $A, B, C, \ldots$
- *Líneas*. $l, m, n, \ldots$

Geometría euclídea plana

Nociones fundamentales

- *Puntos*. $A, B, C, \dots$
- *Líneas*. $l, m, n, \dots$
- *incidencia*. $A \sim l$
- *orden*. $A * B * C$
- *congruencia de segmentos*. $\overline{AB} \cong \overline{CD}$
- *congruencia de ángulos*. $\angle ABC \cong \angle CDE$

Geometría de incidencia

```
class incidence_geometry (Point Line : Type*) :=
  (lies_on : Point → Line → Prop)
  (infix ` ~ ` : 50 := lies_on)
```

Geometría de incidencia

```
class incidence_geometry (Point Line : Type*) :=
  (lies_on : Point → Line → Prop)
  (infix ` ~ ` : 50 := lies_on)
  (I1 {A B : Point} (h : A ≠ B) : ∃! l : Line, A ~ l ∧ B ~ l)
```

Geometría de incidencia

```
class incidence_geometry (Point Line : Type*) :=
  (lies_on : Point → Line → Prop)
  (infix ` ~ ` : 50 := lies_on)
  (I1 {A B : Point} (h : A ≠ B) : ∃! l : Line, A ~ l ∧ B ~ l)
  (I2 (l : Line) : ∃ A B : Point, A ≠ B ∧ A ~ l ∧ B ~ l)
```

Geometría de incidencia

```
class incidence_geometry (Point Line : Type*) :=
  (lies_on : Point → Line → Prop)
  (infix ` ~ ` : 50 := lies_on)
  (I1 {A B : Point} (h : A ≠ B) : ∃! l : Line, A ~ l ∧ B ~ l)
  (I2 (l : Line) : ∃ A B : Point, A ≠ B ∧ A ~ l ∧ B ~ l)
  (I3 : ∃ A B C : Point, neq3 A B C ∧ ¬ ∃ l : Line,  A ~ l ∧ B ~ l ∧ C ~ l)
```

Geometría de incidencia

Proposición

Dos líneas distintas pueden tener como mucho un punto en común.

Geometría de incidencia

Proposición

Dos líneas distintas pueden tener como mucho un punto en común.

```
def is_common_point (A : Point) (l m : Line) :=
  A ~ l ∧ A ~ m
```

```
def have_common_point (l m : Line) :=
  ∃ A : Point, is_common_point A l m
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
```

```
Point Line: Type u
ig: incidence_geometry Point Line
```

```
⊢ ∀ (l m : Line), l ≠ m →
  (∃! A : Point, is_common_point A l m)
  ∨ ¬ have_common_point Point l m
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
  intros l m
```

```
Point Line: Type u
ig: incidence_geometry Point Line
```

```
⊢ ∀ (l m : Line), l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
  intros l m,
```

```
Point Line: Type u
ig: incidence_geometry Point Line
l m : Line

⊢ l ≠ m →
  (∃! A : Point, is_common_point A l m)
  ∨ ¬ have_common_point Point l m
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
  intros l m,
  contrapose
```

```
Point Line: Type u
ig: incidence_geometry Point Line
l m : Line
```

```
⊢ l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
  intros l m,
  contrapose,
```

```
Point Line: Type u
ig: incidence_geometry Point Line
l m : Line
```

```
⊢ ¬((∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m) →
  ¬ l ≠ m
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
  intros l m,
  contrapose,
  push_neg,
  rintro ⟨not_unique, hlm⟩,
  rw exists_unique at not_unique,
  push_neg at not_unique,
  cases hlm with A hA,
  rcases not_unique A hA with ⟨B, ⟨hB, hAB⟩⟩,
  rw ne_comm at hAB
```

```
Point Line: Type u
ig: incidence_geometry Point Line
l m : Line

⊢ ¬((∃! A : Point, is_common_point A l m)
     ∨ ¬ have_common_point Point l m) →
   ¬ l ≠ m
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
  intros l m,
  contrapose,
  push_neg,
  rintro ⟨not_unique, hlm⟩,
  rw exists_unique at not_unique,
  push_neg at not_unique,
  cases hlm with A hA,
  rcases not_unique A hA with ⟨B, ⟨hB, hAB⟩⟩,
  rw ne_comm at hAB,
```

```
Point Line: Type u
ig: incidence_geometry Point Line
l m : Line
A B: Point
hA: is_common_point A l m
hB: is_common_point B l m
hAB: A ≠ B

⊢ l = m
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
  intros l m,
  contrapose,
  push_neg,
  rintro ⟨not_unique, hlm⟩,
  rw exists_unique at not_unique,
  push_neg at not_unique,
  cases hlm with A hA,
  rcases not_unique A hA with ⟨B, ⟨hB, hAB⟩⟩,
  rw ne_comm at hAB,
  exact unique_of_exists_unique (ig.I1 hAB) ⟨hA.1, hB.1⟩ ⟨hA.2, hB.2⟩
```

```
Point Line: Type u
ig: incidence_geometry Point Line
l m : Line
A B: Point
hA: is_common_point A l m
hB: is_common_point B l m
hAB: A ≠ B

⊢ l = m
```

```
ig.I1 hAB : ∃! l : Line, A ~ l ∧ B ~ l
```

Geometría de incidencia

```
lemma neq_lines_have_at_most_one_common_point :
  ∀ l m : Line, l ≠ m →
    (∃! A : Point, is_common_point A l m)
    ∨ ¬ have_common_point Point l m :=
begin
  intros l m,
  contrapose,
  push_neg,
  rintro ⟨not_unique, hlm⟩,
  rw exists_unique at not_unique,
  push_neg at not_unique,
  cases hlm with A hA,
  rcases not_unique A hA with ⟨B, ⟨hB, hAB⟩⟩,
  rw ne_comm at hAB,
  exact unique_of_exists_unique (ig.I1 hAB) ⟨hA.1, hB.1⟩ ⟨hA.2, hB.2⟩
end
```

goals accomplished ✓

Geometría del orden

Definición

Dados dos puntos distintos A, B el **segmento** $\overline{AB}$ es el *conjunto de puntos* que contiene a A, B y a todos los puntos que están entre ellos.

Geometría del orden

Definición

Dos puntos distintos A, B determinan el **segmento** $\overline{AB}$.

```
structure Seg := {A B : Point} (neq : A ≠ B)
```

Geometría del orden

Definición

Dos puntos distintos A, B determinan el **segmento** $\overline{AB}$.

```
structure Seg := {A B : Point} (neq : A ≠ B)
```

Definición

Un punto C **pertenece** al segmento $\overline{AB}$ si coincide con A o B o está entre ellos ($A * C * B$).

```
def Seg.in (seg : Seg) (P : Point) : Prop :=
  P = seg.A ∨ P = seg.B ∨ (og.between seg.A P seg.B)
```

Independencia del axioma de las paralelas

```
def parallel (Point : Type*) {Line : Type*} [ig : incidence_geometry Point Line]
  (l m : Line) : Prop := ¬ ∃ P : Point, is_common_point P l m
```

Independencia del axioma de las paralelas

```
def parallel (Point : Type*) {Line : Type*} [ig : incidence_geometry Point Line]
  (l m : Line) : Prop := ¬ ∃ P : Point, is_common_point P l m

def P (Point Line : Type*) [ig : incidence_geometry Point Line] :=
  ∀ (l : Line) (A : Point), ¬ A ~ l → ∃! m : Line, A ~ m ∧ parallel Point l m
```

Independencia del axioma de las paralelas

```
def parallel (Point : Type*) {Line : Type*} [ig : incidence_geometry Point Line]
  (l m : Line) : Prop := ¬ ∃ P : Point, is_common_point P l m

def P (Point Line : Type*) [ig : incidence_geometry Point Line] :=
  ∀ (l : Line) (A : Point), ¬ A ~ l → ∃! m : Line, A ~ m ∧ parallel Point l m



theorem parallels_independence :
  ¬ ∀ plane : hilbert_plane Point Line,
      P Point Line :=
begin
  sorry
end
```

Independencia del axioma de las paralelas

```
def parallel (Point : Type*) {Line : Type*} [ig : incidence_geometry Point Line]
  (l m : Line) : Prop := ¬ ∃ P : Point, is_common_point P l m

def P (Point Line : Type*) [ig : incidence_geometry Point Line] :=
  ∀ (l : Line) (A : Point), ¬ A ~ l → ∃! m : Line, A ~ m ∧ parallel Point l m
```

```
theorem parallels_independence :
  ¬ ∀ plane : hilbert_plane Point Line,
      P Point Line :=
begin
  sorry
end
```

```
Point Line : Type u
⊢ ¬ ∀ plane : hilbert_plane Point Line,
      P Point Line
```

Independencia del axioma de las paralelas

```
def parallel (Point : Type*) {Line : Type*} [ig : incidence_geometry Point Line]
  (l m : Line) : Prop := ¬ ∃ P : Point, is_common_point P l m

def P (Point Line : Type*) [ig : incidence_geometry Point Line] :=
  ∀ (l : Line) (A : Point), ¬ A ~ l → ∃! m : Line, A ~ m ∧ parallel Point l m
```

```
theorem parallels_independence :
  ¬ ∀ plane : hilbert_plane Point Line,
      P Point Line :=
begin
  push_neg,
  sorry
end
```

```
Point Line : Type u
⊢ ∃ plane : hilbert_plane Point Line,
      ¬ P Point Line
```

Conclusiones

- Cumplimiento del proyecto de formalización de Hilbert.

Conclusiones

- Cumplimiento del proyecto de formalización de Hilbert.
- Lean como herramienta de estudio.

Conclusiones

- Cumplimiento del proyecto de formalización de Hilbert.
- Lean como herramienta de estudio.
- Dificultades con *mathlib*, en particular al intentar definir modelos.

Conclusiones

- Cumplimiento del proyecto de formalización de Hilbert.
- Lean como herramienta de estudio.
- Dificultades con *mathlib*, en particular al intentar definir modelos.
- Queda mucha geometría por formalizar en Lean.

¡Gracias!

Detalle de *La escuela de Atenas*, de Rafael Sanzio, donde vemos a Euclides enseñando a sus alumnos.