Formalizando formas modulares, series de Eisenstein y la declaracion de la conjetura de modularidad.

Christopher Birkbeck

Heilbronn research fellow at University College London

April 20, 2022

¿Qué es Lean?

¿Qué es Lean?

- Lean es un sistema interactivo para la demostración de teoremas, basado en la teoria de tipos.

¿Qué es Lean?

- Lean es un sistema interactivo para la demostración de teoremas, basado en la teoria de tipos.
- Lean es un lenguaje computacional donde uno puede codificar definiciones, teoremas y demostraciones.
- Hay otros programas similares, como por ejemplo Coq, Isabelle/HOL, etc. (Pero no sé nada de ellos).

Usos y objetivos de Lean

Usos y objetivos de Lean

- Lean puede verificar si una demostración tiene errores o esta incompleta.

Usos y objetivos de Lean

- Lean puede verificar si una demostración tiene errores o esta incompleta.
- Crear una biblioteca digital de las matemáticas (en Lean esta lleva el nombre de `mathlib`).
- Usos :
  1. Una biblioteca unificada fácil de navegar.
  2. Verificar que las demostraciones no tienen errores.
  3. Simplificar el proceso de arbitraje.
  4. Formalizar la investigación actual.
  5. Usar el aprendizaje de máquinas para demostrar resultados.
  6. Educación.

Antes de todo esto, necesitamos crear esta biblioteca.

- Actualmente `mathlib` tiene alrededor de $850k$ lineas de código con 35558 definiciones y 85052 teoremas escritos por 238 contribuidores.
- La gran mayoría son resultados "básicos" pero hay proyectos más avanzados como por ejemplo el "Liquid Tensor Experiment" en donde están formalizado resultados recientes de Clausen–Scholze.
- El aprendizaje de máquinas ya ha dado resultados interesantes en Lean. Por ejemplo, dada una lista de problemas de las olimpiadas matemáticas, han podido aprender como resolver otros problemas.

- Hoy quiero demostrar como es este proceso usando el ejemplo de las formas modulares, series de Eisenstein y la declaración de la conjetura de modularidad.

- Hoy quiero demostrar como es este proceso usando el ejemplo de las formas modulares, series de Eisenstein y la declaración de la conjetura de modularidad.
- Comencemos viendo la primera definición que queremos formalizar.

Formas modulares

Definicion

Una forma modular es una función $f : \mathbb{H} \to \mathbb{C}$ tal que :

1. Dado un entero $k \in \mathbb{Z}$ (el peso) y un subgrupo $\Gamma \subseteq SL_2(\mathbb{Z})$ (el nivel) esta función satisface : Para cualquier $\gamma = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \Gamma$, tenemos

$$f\left(\frac{az+b}{cz+d}\right) = (cz+d)^k f(z).$$

   Si escribimos $f \mid_k \gamma = (cz+d)^{-k} f(\gamma \cdot z)$, podemos reformular esta propiedad como $f \mid_k \gamma = f$ para cualquier $\gamma \in \Gamma$. Tales funciones se llaman débilmente modulares.

2. f es holomorfa sobre $\mathbb{H}$.

3. f es "acotada en el infinito", significando que para cualquier $\gamma \in \mathrm{SL}_2(\mathbb{Z})$, existen $A, B \in \mathbb{R}$ tal que para cualquier $z \in \mathbb{H}$, con $A \leq \mathrm{Im}(z)$, tenemos $|(f \mid_k \gamma)(z)| \leq B$.

Si en vez de (3) insistimos que:

> Para cualquier $\gamma \in \mathrm{SL}_2(\mathbb{Z})$ y $0 < \epsilon$, existe $A \in \mathbb{R}$ tal que para cualquier $z \in \mathbb{H}$, con $A \leq \mathrm{Im}(z)$, tenemos $|(f \mid_k \gamma)(z)| \leq \epsilon$.

Las funciones que resultan son llamadas "cero en el infinito", dándonos formas modulares cuspidales. Si escogemos k, Γ adecuadamente, esta construcción produce espacios vectoriales (no triviales) de dimensión finita. Los denotamos como $M_k(\Gamma)$ y $S_k(\Gamma)$.

Formas modulares en Lean

Veamos como se traduce esta definición en Lean. Primero que todo, `mathlib` ya tiene una buena base de donde podemos comenzar. Por ejemplo, gracias al trabajo de Alex Kontorovich, Heather Macbeth y Marc Masdeu, ya están formalizadas las definiciones del semiplano positivo y la acción dada por la transformación de Moebious.

Comenzamos definiendo la acción de peso k, llamada "`slash_k`", sobre funciones $f : \mathbb{H} \to \mathbb{C}$. En Lean esto se escribe así:

```
def slash_k : ℤ → GL(2, ℝ)⁺ → (ℍ → ℂ) → (ℍ → ℂ) :=
λ k γ f, (λ (z : ℍ),
f (γ • z) * (γ.1.det)^(k-1) * ((γ 1 0 * z + γ 1 1)^k)⁻¹)
```

Le damos el nombre de $| [k]\gamma$ en lean.

Comenzamos definiendo la acción de peso k, llamada "`slash_k`", sobre funciones $f : \mathbb{H} \to \mathbb{C}$. En Lean esto se escribe así:

```
def slash_k : ℤ → GL(2, ℝ)⁺ → (ℍ → ℂ) → (ℍ → ℂ) :=
λ k γ f, (λ (z : ℍ),
f (γ • z) * (γ.1.det)^(k-1) * ((γ 1 0 * z + γ 1 1)^k)⁻¹)
```

Le damos el nombre de $| [k]\gamma$ en lean.

Dar una demostración en vivo si es posible

Usando esto definimos el espacio vectorial (complejo) de funciones $f : \mathbb{H} \to \mathbb{C}$ que son débilmente modulares:

```
def weakly_modular_submodule --submodulo de formas
    debilmente modulares
(k : ℤ) (Γ : subgroup SL(2,ℤ)): submodule ℂ (ℍ  → ℂ) :=
{carrier := {f : (ℍ → ℂ) |
∀ (γ : Γ),  (f ∣[k] (γ : GL(2, ℝ)⁺)) = f },
zero_mem' := ...,  --zero
add_mem' :=...,    --suma
smul_mem' :=...,} --producto escalar
```

Ahora veamos como formalizar la holomorficidad. (des)Afortunadamente `mathlib` ya tiene varias definiciones:

1. Dada una función $f : \mathbb{C} \to \mathbb{C}$ y S un subconjuto de $\mathbb{C}$, tenemos `differentiable_on` que define cuando f es diferenciable (holomorfa) sobre S. Esta definición tiene ventajas y desventajas:
   - Es una propiedad de funciones desde un espacio vectorial normado, pero $\mathbb{H}$ no es un espacio vectorial. Entonces si queremos usar esta definición para las funciones $f : \mathbb{H} \to \mathbb{C}$ tendremos que convertirlas a funciones $f : \mathbb{C} \to \mathbb{C}$, por ejemplo, extendiendo con cero.
   - Una gran cantidad de resultados de análisis complejo están formalizados usando `differentiable_on`. Por ejemplo, el Teorema integral de Cauchy, que necesitaremos mas tarde.
2. Para funciones $f : M \to M'$ entre dos variedades, podemos usar `mdifferentiable`. Esto define cuando tales funciones son diferenciables, pero ahora uno las considera como funciones entre variedades.

- La segunda opción es la mas agradable pero la primera es mas fácil de usar.
- Por hoy vamos a usar la segunda opción. Para esto tenemos que convertir $\mathbb{H}$ a una variedad compleja. La manera mas fácil de hacer esto es convirtiendo $\mathbb{H}$ a un `termino` de `Tipo` "subconjutos abiertos de $\mathbb{C}$". Para esto uno lo considera como un conjunto que contiene a $S \subseteq \mathbb{C}$ y una demostración que es abierto. Denotamos con $\mathbb{H}'$ a $\mathbb{H}$ pero ahora considerado como un subconjunto abierto.

Finalmente definimos la condición de ser “acotado en el infinito” y “cero en el infinito”.

```
def is_bound_at_infinity := { f : ℍ → ℂ | ∃ (M A : ℝ), ∀
    z : ℍ, A ≤ im z → abs (f z) ≤ M }

def is_zero_at_infinity := { f : ℍ → ℂ | ∀ ε : ℝ, 0 < ε
    → ∃ A : ℝ, ∀ z : ℍ, A ≤ im z  → abs (f z) ≤ ε }
```

Formas modulares en Lean

Uniendo todo tenemos:

```
structure is_modular_form_of_lvl_and_weight
(Γ : subgroup SL(2,ℤ))
(k : ℤ) (f : ℍ → ℂ) : Prop :=
(hol : mdifferentiable 𝒥(ℂ) 𝒥(ℂ) (↑f : ℍ' → ℂ))
(transf :  f ∈ weakly_modular_submodule k Γ )
(infinity : ∀ (A :  ( ⊤ : subgroup SL(2,ℤ))), (f ∣[k] A) ∈
    is_bound_at_infinity )
```

Más aún podemos ver fácilmente que esto define un espacio vectorial complejo:

```
def space_of_mod_forms_of_level_and_weight
(Γ : subgroup SL(2,ℤ)) (k : ℤ): submodule ℂ (ℍ → ℂ):={
carrier:={ f : ℍ → ℂ |
is_modular_form_of_lvl_and_weight Γ k f},...,}
```

Igualmente para la formas cuspidales.

Series de Eisenstein

Los primeros ejemplos interesantes de formas modulares son las series de Eisenstein. Estas son funciones $\mathbb{H} \to \mathbb{C}$ dadas por

$$G_k(z) = \sum_{(c,d)\neq(0,0)} \frac{1}{(cz+d)^k}.$$

Resulta que estas son formas modulares de nivel $\mathrm{SL}_2(\mathbb{Z})$ y si k es par y $k \geq 4$ no son cero.

Series de Eisenstein

Definir las series de Eisenstein en lean es fácil, lo difícil es desmostrar que son formas modulares, en particular demostrar que son holomorfas. Comenzamos con:

```
def Eise (k: ℤ) (z : ℍ) : ℤ × ℤ →  ℂ :=
λ x, 1/(x.1 * z + x.2)^k
```

Luego definimos

```
def Eisenstein_series_of_weight_ (k: ℤ) : ℍ → ℂ :=
λ z, Σ' (x : ℤ × ℤ), (Eise k z x)
```

Acá $\sum'$ no es la suma de $(c, d) \neq (0, 0)$. Esto denota `tsum`, que define sumas infinitas en monoides topológicos. La definición es tal que si la suma es absolutamente convergente da el resultado correcto y si no te da cero.

Series de Eisenstein

Esta definición tiene unas ventajas y desventajas. Una desventaja es que no sirve para definir

$$G_2(z) = \sum_{(c,d)\neq(0,0)} \frac{1}{(cz+d)^2},$$

la cual no es una forma modular, pero todavía es una función interesante. El problema es que no es débilmente modular, dado que no es absolutamente convergente. Pero como `tsum` da cero en este caso, resulta que `Eisenstein_series_of_weight_` es débilmente modular para cualquier $k \in \mathbb{Z}$ (que quizá es una pequeña ventaja).

Series de Eisenstein

La demostración que son débilmente modulares tiene dos pasos:

```
lemma Eise_moeb (k: ℤ) (z : ℍ) (A : SL(2,ℤ)) (i : ℤ × ℤ ):
    Eise k ( (A : GL(2, ℝ)⁺)) · z) i =
((A.1 1 0 * z + A.1 1 1)^k) * (Eise k z (Ind_equiv A i ) )
```

con lo cual tenemos

```
lemma Eisenstein_is_wmodular (Γ : subgroup SL(2,ℤ)) (k : ℤ)
   : (Eisenstein_series_of_weight_ k) ∈
(weakly_modular_submodule k Γ) :=
```

- La demostración no necesita usar que la suma es absolutamente y uniformemente convergente. Pero uno no se escapa, la demostración que es holomorfa viene a cobrar.
- Primero tenemos que reordenar la suma que define G_k. Escribimos G_k como

$$G_k(z) = \sum_{n=0}^{\infty} G_{k,n}(z), \qquad G_{k,n}(z) := \sum_{(c,d)\in S(n)} \frac{1}{(cz+d)^k}$$

  donde $S(n) = \{(c,d) \in \mathbb{Z} \times \mathbb{Z} \mid \max(|c|,|d|) = n\}$.
- La idea es demostrar que esta suma converge absolutamente a G_k y mas aun que la suma $\sum_{n=0}^{N} G_{k,n}$ converge uniformemente a G_k (cuando restringida a ciertas regiones)

La demostración que la suma converge absolutamente es fácil pero fastidiosa. Básicamente uno emplea que $G_{k,n,\mathrm{abs}}(z) \leq 8n^{1-k} r(z)^{-k}$, donde

$$G_{k,n,\mathrm{abs}}(z) := \sum_{(c,d)\in S(n)} \frac{1}{|(cz+d)^k|}$$

y

$$r(x+iy) = \min\left(y, \left(\frac{y^4+(xy)^2}{(x^2+y^2)^2}\right)^{1/2}\right).$$

Dado esto, resulta que

$$|G_k(z)| \leq \sum_n G_{k,n,\mathrm{abs}}(z) \leq 8\zeta(k-1) r(z)^{-k}.$$

Finalmente usando la prueba M de Weierstrass da el resultado.

En lean la convergencia absoluta es la declaración:

```
lemma Eisenstein_series_is_summable (k : ℕ) (z : ℍ)
(h : 3 ≤ k) : summable (Eise k z) :=
```

Para la convergencia uniforme hay que restringir a $\mathbb{H}_{a,b} := \{z = x + iy \in \mathbb{H} \mid |x| \leq a, |y| \geq b\}$ para $a, b \in \mathbb{R}$, con $0 < b$. Aquí tenemos:

```
lemma Eisen_partial_tendsto_uniformly (k : ℤ) (h : 3 ≤ k)
(A B : ℝ) (ha : 0 ≤ A) (hb : 0 < B) :
tendsto_uniformly (Eisen_par_sum_slice k A B)
(Eisenstein_series_restrict k A B) filter.at_top:=
```

Con esto uno demuestra que para $k \geq 3$, G_k es holomorfa usando:

Teorema

Si $\{f_n\}$ es una secuencia de funciones holomorfas sobre un subconjunto abierto S y si sobre cada subconjunto compacto C de S, la secuencia converge uniformemente a una funcion f, entonces f es holomorfa.

Una manera de demostrar esto es usando el teorema integral de Cauchy:

- Dado que cada f_n es holomorfa, para cualquier $x \in S$, podemos escribir

$$f_n(w) = \int_{C_r(x)} \frac{f_n(\zeta)}{\zeta - w} d\zeta = \frac{1}{2\pi i} \int_0^{2\pi} \frac{re^{i\theta} i f_n(x + re^{i\theta})}{(x + re^{i\theta} - w)} d\theta$$

  con $C_r(x)$ un disco suficientemente pequeño alrededor de x (en S), con radio r conteniendo a w.

- Ahora, como f_n converge uniformemente a f, tenemos que

$$f(x) = \int_{C_r(x)} \frac{f(\zeta)}{\zeta - x} d\zeta$$

  y finalmente uno demuestra que f es diferenciable dado que podemos diferenciar dentro de la integral.

- Como la holomorficidad es una propiedad local, basta con usar discos en lugar de S.

Entonces queremos primero demostrar que si f_n converge uniformemente a f, entonces $\int f_n$ converge a $\int f$.

```
lemma int_uniform_lim_eq_lim_of_int (R : ℝ) (hR: 0 < R)
(F : ℕ → ℂ → ℂ) (f : ℂ → ℂ) (z : ℂ)  (w : ball z R)
(F_cts : ∀ n, continuous_on (F n) (closed_ball z R))
(hlim : tendsto_uniformly_on F f filter.at_top
(closed_ball z R) ) :
tendsto
(λ n, ∫ (θ : ℝ) in 0..2 * π,
(cauchy_disk_function R z (F n) w) θ)
at_top
(𝓝 (∫ (θ : ℝ) in 0..2 * π,
*(cauchy_disk_function R z f w) θ)) :=
```

aquí `cauchy_disk_function f` denota la función $\theta \mapsto \frac{1}{2\pi i} \frac{ire^{i\theta} f(z+re^{i\theta})}{(z+re^{i\theta}-w)}$

Tambien necesitamos demostrar que la función definida por la integral es diferenciable:

```
lemma cauchy_disk_form_differentiable_on (R r: ℝ)
(hR: 0 < R) (hr : r < R) (hr' : 0 ≤ r) (z : ℂ)
(f : ℂ → ℂ) (hf : continuous_on f (closed_ball z R)) :
differentiable_on ℂ (cauchy_disk_form R z f) (ball z r) :=
```

aquí `cauchy_disk_form f` denota la función $\mathbb{C} \to \mathbb{C}$ dada por $w \mapsto \int_{C_R(z)} \frac{f(\zeta)}{\zeta - w} d\zeta$.

Combinando todo tenemos:

```
lemma unif_of_diff_is_diff (F : ℕ → ℂ → ℂ) (f : ℂ → ℂ)
(z : ℂ) (R r : ℝ) (hR : 0 < R) (hr : r < R) (hr' : 0 ≤ r)
(hdiff : ∀ (n : ℕ), differentiable_on ℂ (F n)
(closed_ball z R))
(hlim : tendsto_uniformly_on F f filter.at_top
(closed_ball z R)): differentiable_on ℂ f (ball z r) :=
```

Este fue el mayor resultado necesario para este proyecto. Con esto es fácil demostrar que:

```
lemma Eisenstein_is_mdiff (k : ℕ) (hk : 3 ≤ k) :
mdifferentiable 𝒥(ℂ) 𝒥(ℂ) (↑(Eisenstein_series_of_weight_ k) : ℍ' → ℂ) :=
```

Acotado en el infinito

Finalmente necesitamos demostrar que estas funciones son acotadas en el infinito. La demostración requiere saber que estas funciones son débilmente modulares y absolutamente convergentes:

```
lemma Eisenstein_is_bounded (k : ℕ) (hk : 3 ≤ k) :
Eisenstein_series_of_weight_ k ∈ is_bound_at_infinity :=
```

```
lemma Eisenstein_series_is_modular_form (k : ℕ) (hk : 3 ≤ k) :
is_modular_form_of_lvl_and_weight ( ⊤ : subgroup SL(2,ℤ)) k
(Eisenstein_series_of_weight_ k) :={

hol:= by {apply Eisenstein_is_mdiff k hk},

transf := by {apply Eisenstein_is_wmodular ( ⊤ : subgroup SL(2,ℤ)) k},

infinity := by {intros A,
rw (wmodular_mem k ( ⊤: subgroup SL(2,ℤ)) (Eisenstein_series_of_weight_ k )).1
(Eisenstein_is_wmodular ( ⊤ : subgroup SL(2,ℤ)) k) A,
apply Eisenstein_is_bounded k hk}}
```

¿Qué falta?

- ¡Demostrar que no son idénticamente cero!
- Ejemplos de formas cuspidales.
- Teoria de q-expansiones.
- Demostrar resultados mas interesantes: teoria Atkin–Lehner, multiplicidad uno, etc.
- Demostrar el teorema de modularidad y FLT...

Conjetura de modularidad

Finalmente veamos como uno formaliza la declaración de la conjetura de modularidad. Primero que todo, esta es la definición actual de una curva elíptica en `mathlib`:

```
structure EllipticCurve (R : Type*) [comm_ring R] :=
(a1 a2 a3 a4 a6 : R)
(disc_unit : R^×)
(disc_unit_eq :
(disc_unit : R) = EllipticCurve.disc_aux a1 a2 a3 a4 a6)
```

Entonces tenemos que hacer esto de la manera mas básica posible. Primero definimos $a_p(E)$.

Definicion

Si $E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ es una curva elíptica y p un numero primo, definimos $n_p(E)$ como el numero de soluciones de $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ en $\mathbb{F}_p$. Con esto definimos $a_p(E) := p - n_p(E)$.

Conjetura de modularidad

Teorema (Conjetura Shimura–Taniyama–Weil : versión a_p)

Dada E, una curva elíptica sobre $\mathbb{Q}$. Existe $N \in \mathbb{N}$ y $f \in M_2(\Gamma_0(N))$ tal que para cualquier numero primo p con $p \nmid N$, tenemos

$$a_p(E) = a_p(f)$$

donde $f = \sum_n a_n(f) q^n$ es la expansión q de f.

Se darán cuenta que nunca hablé de la expansión q de una forma modular en lean, pero uno puede hacer trampa. Hay un resultado que dice:

Lema

Si $f \in M_k(\Gamma_0(N))$ tiene expansión q dada por $\sum_n a_n(f)q^n$, entonces

$$a_n(f) = \int_0^1 f(x+i)e^{-2\pi in(x+i)}dx$$

Entonces podemos definir $a_n(f)$ en lean como :

```
def modular_form_an {N : ℕ} {k : ℤ} (n : ℕ)
(f : space_of_mod_forms_of_level_and_weight (Gamma0_N N) k) : ℂ :=
∫ (x : ℝ) in 0..1, ( exp (-2 * π * I * n *(x + I))) * f.1 (map_to_upper x)
```

donde aquí `map_to_upper` lleva $x \in \mathbb{R}$ a $x + i \in \mathbb{H}$.

Por otro lado, tenemos:

```
def EllipticCurve.ap (E : EllipticCurve ℚ) (p : ℕ) : ℕ :=
p-(cardinal.mk (set_of_points_mod_n E p)).to_nat
```

donde

```
def set_of_points_mod_n (E : EllipticCurve ℚ) (n : ℕ) :=
{P : (zmod n) × (zmod n) | let ⟨x, y⟩ := P in  y^2 + (rat_red E.a1 n) * x * y+ (
    rat_red E.a3 n) * y   =
x^3 +(rat_red E.a2 n) * x^2 + (rat_red E.a4 n) * x + (rat_red E.a6 n)}
```

Conjetura de modularidad

```
theorem modularity_conjecture (E : EllipticCurve ℚ) :
∃ (N : ℕ) (f : space_of_mod_forms_of_level_and_weight (Gamma0_N N) 2),
∀ (p : ℕ) (hp : p.prime ) (hN : (N : zmod p ) ≠ 0 ),
modular_form_an p f = E.ap p :=
begin
sorry,
end
```