Carnegie Mellon University

Large-Scale Formalisation: A Student's Perspective

Thematic Program on AI and Mathematics, Korea Institute for Advanced Study

---

Sidharth Hariharan
based on joint work with Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska, and community contributors

13 July, 2026

sidharthhariharan@cmu.edu

Contents

The Beginnings of a Formalisation Project

Why (do I like to) formalise?

Why (do I like to) formalise?

In undergrad, I had a very interesting professor who wears even more interesting trousers.

Why (do I like to) formalise?

In undergrad, I had a very interesting professor who wears even more interesting trousers. He once told me,

Why (do I like to) formalise?

In undergrad, I had a very interesting professor who wears even more interesting trousers. He once told me,

> “Formalising mathematics is a great way to learn it.”

Why (do I like to) formalise?

In undergrad, I had a very interesting professor who wears even more interesting trousers. He once told me,

> "Formalising mathematics is a great way to learn it."

I like learning mathematics.

Why (do I like to) formalise?

In undergrad, I had a very interesting professor who wears even more interesting trousers. He once told me,

"Formalising mathematics is a great way to learn it."

I like learning mathematics. I soon discovered that meeting the demands of interactive theorem provers adds very interesting perspectives.

How a Project Usually Goes

How a Project Usually Goes

1. A blueprint is written.

How a Project Usually Goes

1. A blueprint is written.
2. Basic definitions are formalised, with the aim of stating the main result.

How a Project Usually Goes

1. A blueprint is written.
2. Basic definitions are formalised, with the aim of stating the main result.
3. The dependency graph is then traversed*.

How a Project Usually Goes

1. A blueprint is written.
2. Basic definitions are formalised, with the aim of stating the main result.
3. The dependency graph is then traversed*.
4. As the formalisation progresses, the blueprint and dependency graph are typically updated, refined, refactored and, if needed, corrected.

How a Project Usually Goes

1. A blueprint is written.
2. Basic definitions are formalised, with the aim of stating the main result.
3. The dependency graph is then traversed*.
4. As the formalisation progresses, the blueprint and dependency graph are typically updated, refined, refactored and, if needed, corrected.
5. In parallel, tools and mechanisms are sometimes developed to ease the traversal.

How a Project Usually Goes

1. A blueprint is written.
2. Basic definitions are formalised, with the aim of stating the main result.
3. The dependency graph is then traversed*.
4. As the formalisation progresses, the blueprint and dependency graph are typically updated, refined, refactored and, if needed, corrected.
5. In parallel, tools and mechanisms are sometimes developed to ease the traversal.

* Traversing the dependency graph is not always the only goal of a formalisation project. An often unspoken goal is to explore the possibility of extending, generalising, reframing, or otherwise modifying the mathematics to make the code more suitable for maintenance and reuse.

Writing a Blueprint

What is a blueprint?

What is a blueprint?

- Think of the blueprint an architect might design before constructing a building.

What is a blueprint?

- Think of the blueprint an architect might design before constructing a building.
- A blueprint guides the effort by fleshing out some of the finer details of the proof that are anticipated to come up during formalisation.

What is a blueprint?

- Think of the blueprint an architect might design before constructing a building.
- A blueprint guides the effort by fleshing out some of the finer details of the proof that are anticipated to come up during formalisation.

What is a blueprint?

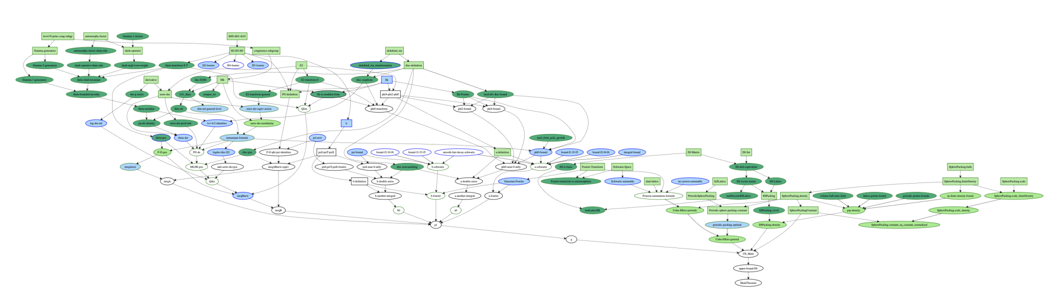
- Think of the blueprint an architect might design before constructing a building.
- A blueprint guides the effort by fleshing out some of the finer details of the proof that are anticipated to come up during formalisation.
- The target audience of a blueprint is formalisers who may not be experts in the mathematics.

What is a blueprint?

- Think of the blueprint an architect might design before constructing a building.
- A blueprint guides the effort by fleshing out some of the finer details of the proof that are anticipated to come up during formalisation.
- The target audience of a blueprint is formalisers who may not be experts in the mathematics.
- A blueprint is typically used to build a dependency graph.

What is a blueprint?

- Think of the blueprint an architect might design before constructing a building.
- A blueprint guides the effort by fleshing out some of the finer details of the proof that are anticipated to come up during formalisation.
- The target audience of a blueprint is formalisers who may not be experts in the mathematics.
- A blueprint is typically used to build a dependency graph.



Sphere Packing Blueprint Timeline

Sphere Packing Blueprint Timeline

- Initial version written by MV based on original paper [5]

Sphere Packing Blueprint Timeline

- Initial version written by MV based on original paper [5]
- First major modification: SL’s original proofs of the modular form inequalities [4]

- Initial version written by MV based on original paper [5]
- First major modification: SL's original proofs of the modular form inequalities [4]
- Second major modification: Library building for sphere packing theory (GM, BM, SH) and modular form theory (CB, SL)

Sphere Packing Blueprint Timeline

- Initial version written by MV based on original paper [5]
- First major modification: SL's original proofs of the modular form inequalities [4]
- Second major modification: Library building for sphere packing theory (GM, BM, SH) and modular form theory (CB, SL)
- Third major modification: Upstreaming parts of SH's MSc thesis [3]

Sphere Packing Blueprint Timeline

- Initial version written by MV based on original paper [5]
- First major modification: SL's original proofs of the modular form inequalities [4]
- Second major modification: Library building for sphere packing theory (GM, BM, SH) and modular form theory (CB, SL)
- Third major modification: Upstreaming parts of SH's MSc thesis [3]

`https://thefundamentaltheor3m.github.io/Sphere-Packing-Lean/blueprint`

Learning through Blueprinting: Constructing the ‘Magic Function’

March 2024 — A modular form is “something something Ramanujan”

Learning through Blueprinting: Constructing the ‘Magic Function’

March 2024	A modular form is “something something Ramanujan”
August 2024	I understand the power of Cohn–Elkies [2, Theorem 3.1]

Learning through Blueprinting: Constructing the 'Magic Function'

March 2024	A modular form is "something something Ramanujan"
August 2024	I understand the power of Cohn–Elkies [2, Theorem 3.1]
October 2024	I understand the 3 main properties of the magic function

Learning through Blueprinting: Constructing the ‘Magic Function’

March 2024	A modular form is “something something Ramanujan”
August 2024	I understand the power of Cohn–Elkies [2, Theorem 3.1]
October 2024	I understand the 3 main properties of the magic function
	I understand the key ingredients needed to prove it has these properties

Learning through Blueprinting: Constructing the ‘Magic Function’

- **March 2024** — A modular form is “something something Ramanujan”
- **August 2024** — I understand the power of Cohn–Elkies [2, Theorem 3.1]
- **October 2024** — I understand the 3 main properties of the magic function
- — I understand the key ingredients needed to prove it has these properties
- — I understand how to prove those ingredients assuming only results in Mathlib, other parts of the sphere packing project, and one very hard contour integral

Learning through Blueprinting: Constructing the 'Magic Function'

March 2024	A modular form is "something something Ramanujan"
August 2024	I understand the power of Cohn–Elkies [2, Theorem 3.1]
October 2024	I understand the 3 main properties of the magic function
	I understand the key ingredients needed to prove it has these properties
	I understand how to prove those ingredients assuming only results in Mathlib, other parts of the sphere packing project, and one very hard contour integral
March 2025	I understand how to use the ingredients to deduce the properties, and how inspecting Cohn–Elkies gives us those properties

Learning through Blueprinting: Constructing the 'Magic Function'

- **March 2024** — A modular form is "something something Ramanujan"
- **August 2024** — I understand the power of Cohn–Elkies [2, Theorem 3.1]
- **October 2024** — I understand the 3 main properties of the magic function
- I understand the key ingredients needed to prove it has these properties
- I understand how to prove those ingredients assuming only results in Mathlib, other parts of the sphere packing project, and one very hard contour integral
- **March 2025** — I understand how to use the ingredients to deduce the properties, and how inspecting Cohn–Elkies gives us those properties
- **June 2025** — **I have a detailed, formalisation-friendly exposition of the construction of the magic function**

The Devil is in the Details

What is a sphere packing, really?

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024):

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024):

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

Second attempt (4 July, 2024):

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

Second attempt (4 July, 2024):

```
def Discrete (X : Set V) : Prop := (Countable X) ∧ ( ∀ x ∈ X, ∃ ε > 0, ∀ y ∈ X, y ≠ x →
     Euclidean.dist x y > ε)
def Centres (X : Set V) : Prop := (Discrete d X) ∧ ( ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist
     x y ≥ 2)
def Packing (X P : Set V) : Prop := (Centres d X) ∧ (P = ⋃ x ∈ X, {y | Euclidean.dist x y <
     1})
def isPacking (P : Set V) : Prop := ∃ X, Packing d X P
def LatticePacking ( Λ P : Set V) : Prop := (Packing d Λ P) ∧ (is_lattice Λ)
def isLatticePacking (P : Set V) : Prop := ∃ Λ, LatticePacking d Λ P
```

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

Second attempt (4 July, 2024): A soup of unbundled predicates!

```
def Discrete (X : Set V) : Prop := (Countable X) ∧ ( ∀ x ∈ X, ∃ ε > 0, ∀ y ∈ X, y ≠ x →
     Euclidean.dist x y > ε)
def Centres (X : Set V) : Prop := (Discrete d X) ∧ ( ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist
     x y ≥ 2)
def Packing (X P : Set V) : Prop := (Centres d X) ∧ (P = ⋃ x ∈ X, {y | Euclidean.dist x y <
     1})
def isPacking (P : Set V) : Prop := ∃ X, Packing d X P
def LatticePacking ( Λ P : Set V) : Prop := (Packing d Λ P) ∧ (is_lattice Λ)
def isLatticePacking (P : Set V) : Prop := ∃ Λ, LatticePacking d Λ P
```

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

Second attempt (4 July, 2024): A soup of unbundled predicates! Centres *still* not recoverable!

```
def Discrete (X : Set V) : Prop := (Countable X) ∧ ( ∀ x ∈ X, ∃ ε > 0, ∀ y ∈ X, y ≠ x →
     Euclidean.dist x y > ε)
def Centres (X : Set V) : Prop := (Discrete d X) ∧ ( ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist
     x y ≥ 2)
def Packing (X P : Set V) : Prop := (Centres d X) ∧ (P = ⋃ x ∈ X, {y | Euclidean.dist x y <
     1})
def isPacking (P : Set V) : Prop := ∃ X, Packing d X P
def LatticePacking ( Λ P : Set V) : Prop := (Packing d Λ P) ∧ (is_lattice Λ)
def isLatticePacking (P : Set V) : Prop := ∃ Λ, LatticePacking d Λ P
```

What is a sphere packing, really?

What is a sphere packing, really?

Third attempt (15 July, 2024):

What is a sphere packing, really?

Third attempt (15 July, 2024):

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
     SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { Λ
     : AddSubgroup V} [DiscreteTopology Λ] [isLattice' Λ] where
  periodic : ∀ x ∈ X, ∀ y ∈ Λ, x + y ∈ X
```

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
     SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { Λ
     : AddSubgroup V} [DiscreteTopology Λ] [isLattice' Λ] where
  periodic : ∀ x ∈ X, ∀ y ∈ Λ, x + y ∈ X
```

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
     SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { Λ
     : AddSubgroup V} [DiscreteTopology Λ] [isLattice' Λ] where
  periodic : ∀ x ∈ X, ∀ y ∈ Λ, x + y ∈ X
```

Fourth attempt (30 July, 2024):

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
     SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { Λ
     : AddSubgroup V} [DiscreteTopology Λ] [isLattice' Λ] where
  periodic : ∀ x ∈ X, ∀ y ∈ Λ, x + y ∈ X
```

Fourth attempt (30 July, 2024):

```
structure SpherePacking (d : ℕ) where
  centers : Set (EuclideanSpace ℝ (Fin d))
  radius : ℝ
  radius_pos : 0 < radius
  centers_dist : Pairwise (radius ≤ dist · · : centers → centers → Prop)
structure PeriodicSpherePacking (d : ℕ) extends SpherePacking d where
  Λ : AddSubgroup (EuclideanSpace ℝ (Fin d))
  Λ_discrete : DiscreteTopology Λ := by infer_instance
  Λ_lattice : IsZlattice ℝ Λ := by infer_instance
  Λ_action : ∀ ⦃ x y ⦄, x ∈ Λ → y ∈ centers → x + y ∈ centers
```

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
    SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { Λ
    : AddSubgroup V} [DiscreteTopology Λ] [isLattice' Λ] where
  periodic : ∀ x ∈ X, ∀ y ∈ Λ, x + y ∈ X
```

Fourth attempt (30 July, 2024): Our first stable definition!

```
structure SpherePacking (d : ℕ) where
  centers : Set (EuclideanSpace ℝ (Fin d))
  radius : ℝ
  radius_pos : 0 < radius
  centers_dist : Pairwise (radius ≤ dist · · : centers → centers → Prop)
structure PeriodicSpherePacking (d : ℕ) extends SpherePacking d where
  Λ : AddSubgroup (EuclideanSpace ℝ (Fin d))
  Λ_discrete : DiscreteTopology Λ := by infer_instance
  Λ_lattice : IsZlattice ℝ Λ := by infer_instance
  Λ_action : ∀ ⦃ x y ⦄, x ∈ Λ → y ∈ centers → x + y ∈ centers
```

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
     SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { Λ
     : AddSubgroup V} [DiscreteTopology Λ] [isLattice' Λ] where
  periodic : ∀ x ∈ X, ∀ y ∈ Λ, x + y ∈ X
```

Fourth attempt (30 July, 2024): Our first stable definition! Thank you Gareth Ma!

```
structure SpherePacking (d : ℕ) where
  centers : Set (EuclideanSpace ℝ (Fin d))
  radius : ℝ
  radius_pos : 0 < radius
  centers_dist : Pairwise (radius ≤ dist · · : centers → centers → Prop)
structure PeriodicSpherePacking (d : ℕ) extends SpherePacking d where
  Λ : AddSubgroup (EuclideanSpace ℝ (Fin d))
  Λ_discrete : DiscreteTopology Λ := by infer_instance
  Λ_lattice : IsZlattice ℝ Λ := by infer_instance
  Λ_action : ∀ ⦃ x y ⦄, x ∈ Λ → y ∈ centers → x + y ∈ centers
```

Defining the Magic Function

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument.

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

$$
\begin{aligned}
a(x) := &\int_{-1}^{i} \phi_0\left(\frac{-1}{z+1}\right) (z+1)^2 \, e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \phi_0\left(\frac{-1}{z-1}\right) (z-1)^2 \, e^{\pi i \|x\|^2 z} \, dz \\
&- 2\int_{0}^{i} \phi_0\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz + 2\int_{i}^{i\infty} \phi_0(z) \, e^{\pi i \|x\|^2 z} \, dz
\end{aligned}
$$

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

$$\begin{aligned} a(x) :=& \int_{-1}^{i} \phi_0\left(\frac{-1}{z+1}\right) (z+1)^2 \; e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \phi_0\left(\frac{-1}{z-1}\right) (z-1)^2 \; e^{\pi i \|x\|^2 z} \, dz \\ &- 2\int_{0}^{i} \phi_0\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz + 2\int_{i}^{i\infty} \phi_0(z) \; e^{\pi i \|x\|^2 z} \, dz \\ b(x) :=& \int_{-1}^{i} \psi_S\left(\frac{-1}{z+1}\right) (z+1)^2 \; e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \psi_S\left(\frac{-1}{z-1}\right) (z-1)^2 \; e^{\pi i \|x\|^2 z} \, dz \\ &- 2\int_{0}^{i} \psi_S\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz - 2\int_{i}^{i\infty} \psi_S(z) \; e^{\pi i \|x\|^2 z} \, dz \end{aligned}$$

a and b have three very special properties: they are **radial and Schwartz**, they are (respectively) **Fourier ± 1-eigenfunctions**, and they have **double zeroes at E_8 lattice points.**

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

$$
\begin{aligned}
a(x) := &\int_{-1}^{i} \phi_0\left(\frac{-1}{z+1}\right) (z+1)^2 \; e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \phi_0\left(\frac{-1}{z-1}\right) (z-1)^2 \; e^{\pi i \|x\|^2 z} \, dz \\
&- 2 \int_{0}^{i} \phi_0\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz + 2 \int_{i}^{i\infty} \phi_0(z) \; e^{\pi i \|x\|^2 z} \, dz \\
b(x) := &\int_{-1}^{i} \psi_S\left(\frac{-1}{z+1}\right) (z+1)^2 \; e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \psi_S\left(\frac{-1}{z-1}\right) (z-1)^2 \; e^{\pi i \|x\|^2 z} \, dz \\
&- 2 \int_{0}^{i} \psi_S\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz - 2 \int_{i}^{i\infty} \psi_S(z) \; e^{\pi i \|x\|^2 z} \, dz \\
g(x) := &\frac{\pi i}{8640} a(x) + \frac{i}{240\pi} b(x)
\end{aligned}
$$

a and b have three very special properties: they are **radial and Schwartz**, they are (respectively) **Fourier ± 1-eigenfunctions**, and they have **double zeroes at E_8 lattice points**.

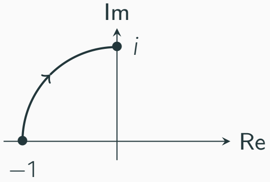
The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

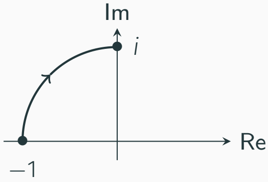
$$
\begin{aligned}
a(x) :=& \int_{-1}^{i} \phi_0\left(\frac{-1}{z+1}\right) (z+1)^2 \; e^{\pi i \|x\|^2 z}\, dz + \int_{1}^{i} \phi_0\left(\frac{-1}{z-1}\right) (z-1)^2 \; e^{\pi i \|x\|^2 z}\, dz \\
&- 2\int_{0}^{i} \phi_0\left(\frac{-1}{z}\right) z^2\, e^{\pi i \|x\|^2 z}\, dz + 2\int_{i}^{i\infty} \phi_0(z)\; e^{\pi i \|x\|^2 z}\, dz \\
b(x) :=& \int_{-1}^{i} \psi_S\left(\frac{-1}{z+1}\right) (z+1)^2 \; e^{\pi i \|x\|^2 z}\, dz + \int_{1}^{i} \psi_S\left(\frac{-1}{z-1}\right) (z-1)^2 \; e^{\pi i \|x\|^2 z}\, dz \\
&- 2\int_{0}^{i} \psi_S\left(\frac{-1}{z}\right) z^2\, e^{\pi i \|x\|^2 z}\, dz - 2\int_{i}^{i\infty} \psi_S(z)\; e^{\pi i \|x\|^2 z}\, dz \\
g(x) :=& \frac{\pi i}{8640} a(x) + \frac{i}{240\pi} b(x)
\end{aligned}
$$

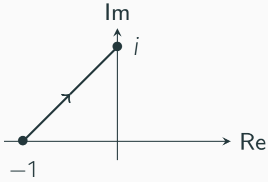
a and b have three very special properties: they are **radial and Schwartz**, they are (respectively) **Fourier ± 1-eigenfunctions**, and they have **double zeroes at E_8 lattice points.** This endows g with unique, 'magic' properties.

Choosing Contours

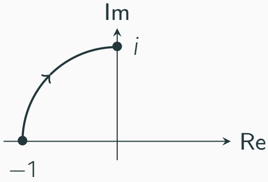
Choosing Contours

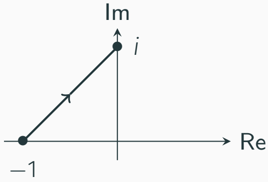


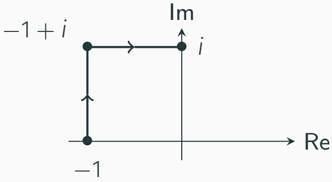

Im
i
Re
−1


Im
i
Re
−1

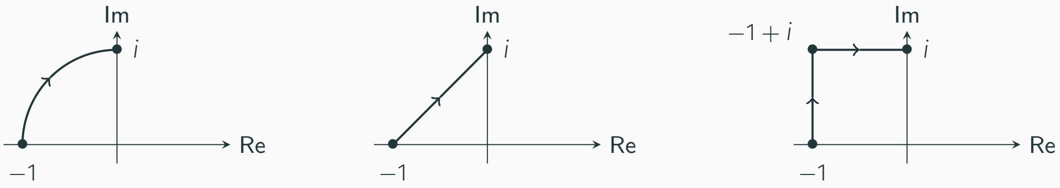
Choosing Contours







Choosing Contours




```
theorem Complex.integral_boundary_rect_eq_zero_of_differentiable_on_off_countable    source
        {E : Type u} [NormedAddCommGroup E] [NormedSpace ℂ E] (f : ℂ → E) (z w : ℂ) (s : Set ℂ)
        (hs : s.Countable) (Hc : ContinuousOn f (Set.uIcc z.re w.re ×ℂ Set.uIcc z.im w.im))
        (Hd :

         ∀ x ∈ Set.Ioo (min z.re w.re) (max z.re w.re) ×ℂ Set.Ioo (min z.im w.im) (max z.im w.im) \ s,
          DifferentiableAt ℂ f x
         )
        :
    (((∫ (x : ℝ) in z.re..w.re, f (↑x + ↑z.im * I)) - ∫ (x : ℝ) in z.re..w.re, f (↑x + ↑w.im * I)) + I •
          ∫ (y : ℝ) in z.im..w.im, f (↑w.re + ↑y * I)) - I • ∫ (y : ℝ) in z.im..w.im, f (↑z.re + ↑y * I)
          = 0
```

Cauchy-Goursat theorem for a rectangle: the integral of a complex differentiable function over the boundary of a rectangle equals zero. More precisely, if `f` is continuous on a closed rectangle and is complex differentiable at all but countably many points of the corresponding open rectangle, then its integral over the boundary of the rectangle equals zero.

Defining the Magic Function in Lean

Defining the Magic Function in Lean

```
def Φ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp ( π * I * r * (z : ℂ))
                   ⋮
def Φ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp ( π * I * r * (z : ℂ))
```

Defining the Magic Function in Lean

```
def Φ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp ( π * I * r * (z : ℂ))
                        ⋮
def Φ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp ( π * I * r * (z : ℂ))
```

```
def Φ₁ : ℝ → ℂ := fun t ↦ I * Φ₁' r (z₁' t)
                       ⋮
def Φ₆ : ℝ → ℂ := fun t ↦ I * Φ₆' r (z₆' t)
```

Defining the Magic Function in Lean

```
def Φ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp ( π * I * r * (z : ℂ))
                    ⋮
def Φ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp ( π * I * r * (z : ℂ))
```

```
def Φ₁ : ℝ → ℂ := fun t ↦ I * Φ₁' r (z₁' t)
                   ⋮
def Φ₆ : ℝ → ℂ := fun t ↦ I * Φ₆' r (z₆' t)
```

```
def I₁' : ℝ → ℂ := fun x ↦ ∫ t in (0 : ℝ)..1, Φ₁ x t
                    ⋮
def I₆' : ℝ → ℂ := fun x ↦ 2 * ∫ t in Ici (1 : ℝ), Φ₆ x t
```

Defining the Magic Function in Lean

```
def Φ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp ( π * I * r * (z : ℂ))
                 ⋮
def Φ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp ( π * I * r * (z : ℂ))
```

```
def Φ₁ : ℝ → ℂ := fun t ↦ I * Φ₁' r (z₁' t)
                ⋮
def Φ₆ : ℝ → ℂ := fun t ↦ I * Φ₆' r (z₆' t)
```

```
def I₁' : ℝ → ℂ := fun x ↦ ∫ t in (0 : ℝ)..1, Φ₁ x t
                 ⋮
def I₆' : ℝ → ℂ := fun x ↦ 2 * ∫ t in Ici (1 : ℝ), Φ₆ x t
```

```
def I₁ : V → ℂ := fun x ↦ I₁' ( ‖ x ‖ ^ 2)
                ⋮
def I₆ : V → ℂ := fun x ↦ I₆' ( ‖ x ‖ ^ 2)
```

Defining the Magic Function in Lean

```
def Φ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp ( π * I * r * (z : ℂ))
                   ⋮
def Φ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp ( π * I * r * (z : ℂ))
```

```
def Φ₁ : ℝ → ℂ := fun t ↦ I * Φ₁' r (z₁' t)
                 ⋮
def Φ₆ : ℝ → ℂ := fun t ↦ I * Φ₆' r (z₆' t)
```

```
def I₁' : ℝ → ℂ := fun x ↦ ∫ t in (0 : ℝ)..1, Φ₁ x t
                   ⋮
def I₆' : ℝ → ℂ := fun x ↦ 2 * ∫ t in Ici (1 : ℝ), Φ₆ x t
```

```
def I₁ : V → ℂ := fun x ↦ I₁' ( ‖ x ‖ ^ 2)
                 ⋮
def I₆ : V → ℂ := fun x ↦ I₆' ( ‖ x ‖ ^ 2)
```

```
def a' : ℝ → ℂ := fun x ↦ I₁' x + I₂' x + I₃' x + I₄' x +
     I₅' x + I₆' x
def a : V → ℂ := fun x ↦ a' ( ‖ x ‖ ^ 2)
```

Defining the Magic Function in Lean

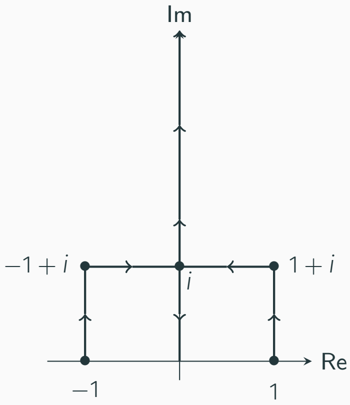
```
def Φ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp ( π * I * r * (z : ℂ))
                  ⋮
def Φ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp ( π * I * r * (z : ℂ))
```

```
def Φ₁ : ℝ → ℂ := fun t ↦ I * Φ₁' r (z₁' t)
                  ⋮
def Φ₆ : ℝ → ℂ := fun t ↦ I * Φ₆' r (z₆' t)
```

```
def I₁' : ℝ → ℂ := fun x ↦ ∫ t in (0 : ℝ)..1, Φ₁ x t
                  ⋮
def I₆' : ℝ → ℂ := fun x ↦ 2 * ∫ t in Ici (1 : ℝ), Φ₆ x t
```

```
def I₁ : V → ℂ := fun x ↦ I₁' ( ‖ x ‖ ^ 2)
                  ⋮
def I₆ : V → ℂ := fun x ↦ I₆' ( ‖ x ‖ ^ 2)
```

```
def a' : ℝ → ℂ := fun x ↦ I₁' x + I₂' x + I₃' x + I₄' x +
     I₅' x + I₆' x
def a : V → ℂ := fun x ↦ a' ( ‖ x ‖ ^ 2)
```



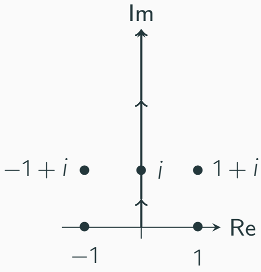
Double Zeroes at Lattice Points

Double Zeroes at Lattice Points

$$-4\sin^2\left(\frac{\pi\left\|x\right\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2\,e^{\pi i\|x\|^2 z}\,dz$$

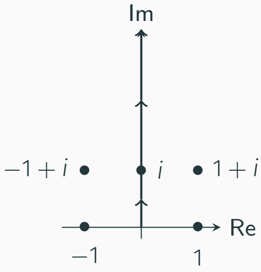
Double Zeroes at Lattice Points

$$-4\sin^2\left(\frac{\pi\|x\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2 e^{\pi i\|x\|^2 z}\,dz$$



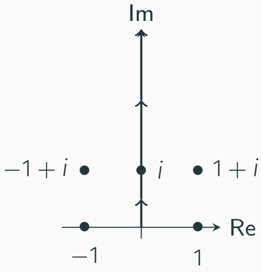
Double Zeroes at Lattice Points

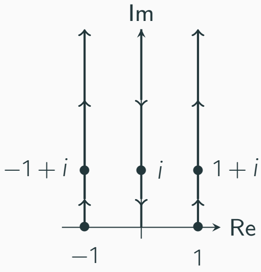
$$-4\sin^2\left(\frac{\pi\|x\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2\,e^{\pi i\|x\|^2 z}\,dz=\int_{-1}^{-1+i\infty}+\int_{1}^{1+i\infty}-2\int_0^{i\infty}$$



Double Zeroes at Lattice Points

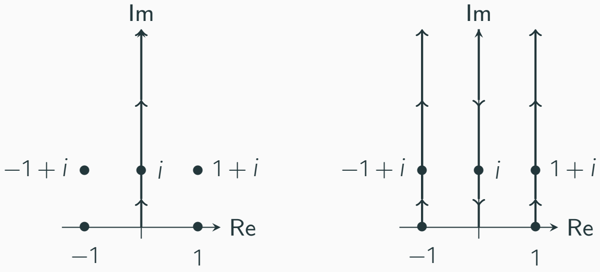
$$-4\sin^2\left(\frac{\pi\|x\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2\,e^{\pi i\|x\|^2 z}\,dz=\int_{-1}^{-1+i\infty}+\int_{1}^{1+i\infty}-2\int_0^{i\infty}$$



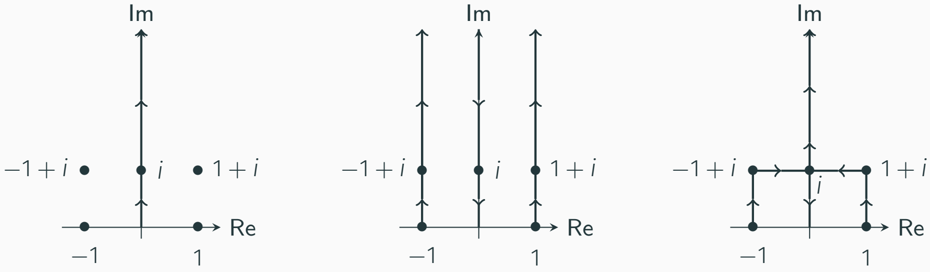


Double Zeroes at Lattice Points

$$-4\sin^2\left(\frac{\pi\|x\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2 e^{\pi i\|x\|^2 z}\,dz = \int_{-1}^{-1+i\infty} + \int_{1}^{1+i\infty} - 2\int_0^{i\infty} = \int_{-1}^{i} + \int_{1}^{i} - 2\int_0^{i} + 2\int_i^{i\infty}$$

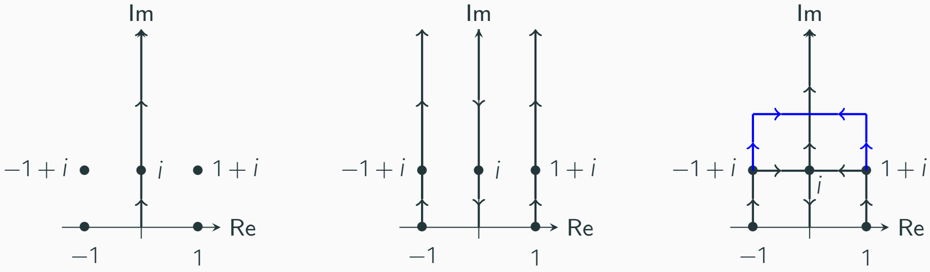


$$-4\sin^2\left(\frac{\pi\,\|x\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2\,e^{\pi i\|x\|^2 z}\,dz=\int_{-1}^{-1+i\infty}+\int_{1}^{1+i\infty}-2\int_0^{i\infty}=\int_{-1}^{i}+\int_{1}^{i}-2\int_0^{i}+2\int_i^{i\infty}$$



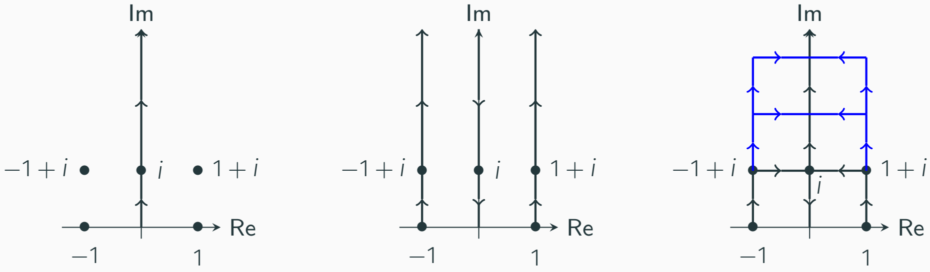
Double Zeroes at Lattice Points

$$-4\sin^2\left(\frac{\pi\left\|x\right\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2\,e^{\pi i\left\|x\right\|^2 z}\,dz=\int_{-1}^{-1+i\infty}+\int_{1}^{1+i\infty}-2\int_0^{i\infty}=\int_{-1}^{i}+\int_{1}^{i}-2\int_0^{i}+2\int_i^{i\infty}$$



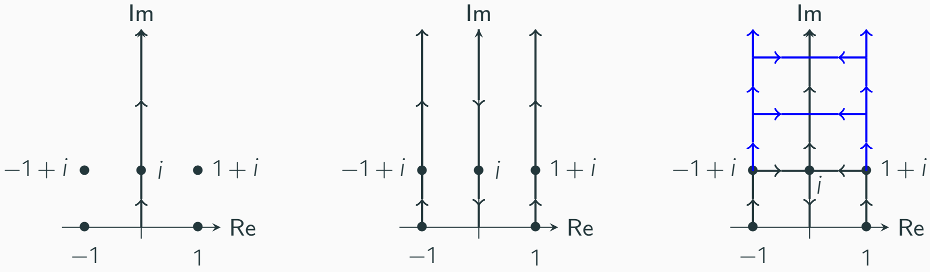
Double Zeroes at Lattice Points

$$-4\sin^2\left(\frac{\pi\,\|x\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2\,e^{\pi i\|x\|^2 z}\,dz=\int_{-1}^{-1+i\infty}+\int_{1}^{1+i\infty}-2\int_0^{i\infty}=\int_{-1}^{i}+\int_{1}^{i}-2\int_0^{i}+2\int_i^{i\infty}$$



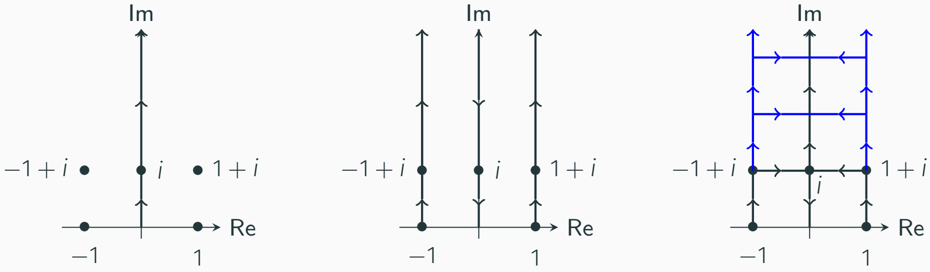
Double Zeroes at Lattice Points

$$-4\sin^2\left(\frac{\pi\,\|x\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2\,e^{\pi i\|x\|^2 z}\,dz=\int_{-1}^{-1+i\infty}+\int_{1}^{1+i\infty}-2\int_0^{i\infty}=\int_{-1}^{i}+\int_{1}^{i}-2\int_0^{i}+2\int_i^{i\infty}$$



Double Zeroes at Lattice Points

$$-4\sin^2\left(\frac{\pi\,\|x\|^2}{2}\right)\int_0^{i\infty}\phi_0\left(\frac{-1}{z}\right)z^2\,e^{\pi i\|x\|^2 z}\,dz = \int_{-1}^{-1+i\infty}+\int_{1}^{1+i\infty}-2\int_0^{i\infty} = \int_{-1}^{i}+\int_{1}^{i}-2\int_0^{i}+2\int_i^{i\infty} = a(x)$$



Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough.

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

There are two ways we can define I_k and J_k:

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

There are two ways we can define I_k and J_k:

1. We can define I'_k and J'_k in terms of r^2 and compose with $\|\cdot\|$ to obtain I_k and J_k.

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

There are two ways we can define I_k and J_k:

1. We can define I'_k and J'_k in terms of r^2 and compose with $\|\cdot\|$ to obtain I_k and J_k.
2. We can define I'_k and J'_k in terms of r and compose with $\|\cdot\|^2$ to obtain I_k and J_k.

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

There are two ways we can define I_k and J_k:

1. We can define I_k' and J_k' in terms of r^2 and compose with $\|\cdot\|$ to obtain I_k and J_k.
2. We can define I_k' and J_k' in terms of r and compose with $\|\cdot\|^2$ to obtain I_k and J_k.

It is quite simple to prove that composing with $\|\cdot\|^2$ preserves Schwartzness.

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

There are two ways we can define I_k and J_k:

1. We can define I'_k and J'_k in terms of r^2 and compose with $\|\cdot\|$ to obtain I_k and J_k.
2. We can define I'_k and J'_k in terms of r and compose with $\|\cdot\|^2$ to obtain I_k and J_k.

It is quite simple to prove that composing with $\|\cdot\|^2$ preserves Schwartzness. However, I'_k and J'_k do not decay on negative inputs if defined in terms of r.

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

There are two ways we can define I_k and J_k:

1. We can define I_k' and J_k' in terms of r^2 and compose with $\|\cdot\|$ to obtain I_k and J_k.
2. We can define I_k' and J_k' in terms of r and compose with $\|\cdot\|^2$ to obtain I_k and J_k.

It is quite simple to prove that composing with $\|\cdot\|^2$ preserves Schwartzness. However, I_k' and J_k' do not decay on negative inputs if defined in terms of r. Here are a few solutions I considered.

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

There are two ways we can define I_k and J_k:

1. We can define I_k' and J_k' in terms of r^2 and compose with $\|\cdot\|$ to obtain I_k and J_k.
2. We can define I_k' and J_k' in terms of r and compose with $\|\cdot\|^2$ to obtain I_k and J_k.

It is quite simple to prove that composing with $\|\cdot\|^2$ preserves Schwartzness. However, I_k' and J_k' do not decay on negative inputs if defined in terms of r. Here are a few solutions I considered.

1. One idea was to use option 1 and prove well-behavedness at 0.

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

There are two ways we can define I_k and J_k:

1. We can define I'_k and J'_k in terms of r^2 and compose with $\|\cdot\|$ to obtain I_k and J_k.
2. We can define I'_k and J'_k in terms of r and compose with $\|\cdot\|^2$ to obtain I_k and J_k.

It is quite simple to prove that composing with $\|\cdot\|^2$ preserves Schwartzness. However, I'_k and J'_k do not decay on negative inputs if defined in terms of r. Here are a few solutions I considered.

1. One idea was to use option 1 and prove well-behavedness at 0.
2. Another idea was to use option 2, define a more general property (“Schwartz on a subset”) and study compositions preserving that property and relating it to Schwartzness.

Proving Schwartzness

Fairly early on in the argument, we end up having to prove that g is a Schwartz function. This is done by showing that the constituent integrals I_k (making up a) and J_k (making up b) are Schwartz.

It is taken for granted in the original paper that proving Schwartzness in $\|x\|$ is enough. But this is quite a nontrivial fact.

There are two ways we can define I_k and J_k:

1. We can define I'_k and J'_k in terms of r^2 and compose with $\|\cdot\|$ to obtain I_k and J_k.
2. We can define I'_k and J'_k in terms of r and compose with $\|\cdot\|^2$ to obtain I_k and J_k.

It is quite simple to prove that composing with $\|\cdot\|^2$ preserves Schwartzness. However, I'_k and J'_k do not decay on negative inputs if defined in terms of r. Here are a few solutions I considered.

1. One idea was to use option 1 and prove well-behavedness at 0.
2. Another idea was to use option 2, define a more general property ("Schwartz on a subset") and study compositions preserving that property and relating it to Schwartzness.
3. Matt Cushman proposed a third idea: to use option 2 but multiply I'_k and J'_k by a smooth transition function. I am currently implementing this.

What Truly Makes a Formalisation

More than just Confidence

More than just Confidence

The most widely advertised reason to formalise is that it gives you **confidence in correctness.**

More than just Confidence

The most widely advertised reason to formalise is that it gives you **confidence in correctness.** What I've learnt is that there is much more to the story.

More than just Confidence

The most widely advertised reason to formalise is that it gives you **confidence in correctness.** What I've learnt is that there is much more to the story.

The very interesting professor was right that formalisation forces one to learn the material:

More than just Confidence

The most widely advertised reason to formalise is that it gives you **confidence in correctness.** What I've learnt is that there is much more to the story.

The very interesting professor was right that formalisation forces one to learn the material: interactive theorem provers demand a great deal of detail.

More than just Confidence

The most widely advertised reason to formalise is that it gives you **confidence in correctness.** What I've learnt is that there is much more to the story.

The very interesting professor was right that formalisation forces one to learn the material: interactive theorem provers demand a great deal of detail. But they also demand the **organisation of the ingredients** making up a result.

More than just Confidence

The most widely advertised reason to formalise is that it gives you **confidence in correctness.** What I've learnt is that there is much more to the story.

The very interesting professor was right that formalisation forces one to learn the material: interactive theorem provers demand a great deal of detail. But they also demand the **organisation of the ingredients** making up a result.

In some sense, "formalisation makes a result bigger than itself."

Modularity

Sphere Packing Theory **(**SH, GM, BM**)**

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Modularity

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

Modularity

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

- Definitions and API for a, b, g, their integrands and parametrisations
- Proving `PolyFourierCoeffBound`, a tool to analyse asymptotic behaviour
- Systematically bounding the integrands of a
- Proving slash action relations for ψ_I, ψ_S, ψ_T
- Cauchy-Goursat for unbounded rectangles

Modularity

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

- Definitions and API for a, b, g, their integrands and parametrisations
- Proving `PolyFourierCoeffBound`, a tool to analyse asymptotic behaviour
- Systematically bounding the integrands of a
- Proving slash action relations for ψ_I, ψ_S, ψ_T
- Cauchy-Goursat for unbounded rectangles

Quasimodular Forms and Inequalities (SL)

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

- Definitions and API for a, b, g, their integrands and parametrisations
- Proving `PolyFourierCoeffBound`, a tool to analyse asymptotic behaviour
- Systematically bounding the integrands of a
- Proving slash action relations for ψ_I, ψ_S, ψ_T
- Cauchy-Goursat for unbounded rectangles

Quasimodular Forms and Inequalities (SL)

- Serre derivatives: definitions and API, equivariance and invariance
- Serre derivatives of E_2, E_4, E_6
- Defining F and G, restriction to the imaginary axis, positivity of restrictions
- Modular linear differential equations

Metaprogramming

A `norm_num` extension for $\mathbb{C}$

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

Metaprogramming

A `norm_num` extension for ℂ

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in ℂ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

Metaprogramming

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

- Developed mainly by Cameron Freer with inputs from Seewoo Lee and Bhavik Mehta

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

- Developed mainly by Cameron Freer with inputs from Seewoo Lee and Bhavik Mehta
- Proves `Tendsto (fun z => expr(f₁ z, ..., fₙ z)) l (nhds c)` where `Tendsto fᵢ l (nhds aᵢ)` are known and the expression is continuous

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

- Developed mainly by Cameron Freer with inputs from Seewoo Lee and Bhavik Mehta
- Proves `Tendsto (fun z => expr(f₁ z, ..., f ₙ z)) l (nhds c)` where `Tendsto f ᵢ l (nhds a ᵢ)` are known and the expression is continuous

```
-- Two atoms, polynomial
example (h₁ : Tendsto f atTop (nhds 0)) (h₂ :
    Tendsto g atTop (nhds 1)) :
   Tendsto (fun z => f z ^ 2 + f z * g z + g
    z ^ 2) atTop (nhds 1) := by tendsto_cont

-- Unused hypotheses don't interfere
example (h₁ : Tendsto f atTop (nhds 0)) (h₂ :
    Tendsto g atTop (nhds 1))
   (unrelated : Tendsto f atBot (nhds 5)) :
   Tendsto (fun z => f z * g z) atTop (nhds
    0) := by tendsto_cont
```

In “Mathematics and the Formal Turn” [1], Avigad effectively presents 18 positives of the advent of formal methods.

The Virtues of Formalisation

In “Mathematics and the Formal Turn” [1], Avigad effectively presents 18 positives of the advent of formal methods.

The first four were both my greatest motivators: **verifying theorems**, **correcting mistakes** (in my understanding), **gaining insight**, and **building libraries**.

The Virtues of Formalisation

In “Mathematics and the Formal Turn” [1], Avigad effectively presents 18 positives of the advent of formal methods.

The first four were both my greatest motivators: **verifying theorems**, **correcting mistakes** (in my understanding), **gaining insight**, and **building libraries**.

The fifth (**searching for definitions and tools**) and fourteenth (**improving access**) greatly benefitted me in my journey.

The Virtues of Formalisation

In "Mathematics and the Formal Turn" [1], Avigad effectively presents 18 positives of the advent of formal methods.

The first four were both my greatest motivators: **verifying theorems**, **correcting mistakes** (in my understanding), **gaining insight**, and **building libraries**.

The fifth (**searching for definitions and tools**) and fourteenth (**improving access**) greatly benefitted me in my journey.

The eighth (**engineering concepts**), ninth (**communicating**), and tenth (**collaborating**) have been key outcomes.

The Virtues of Formalisation

In “Mathematics and the Formal Turn” [1], Avigad effectively presents 18 positives of the advent of formal methods.

The first four were both my greatest motivators: **verifying theorems**, **correcting mistakes** (in my understanding), **gaining insight**, and **building libraries**.

The fifth (**searching for definitions and tools**) and fourteenth (**improving access**) greatly benefitted me in my journey.

The eighth (**engineering concepts**), ninth (**communicating**), and tenth (**collaborating**) have been key outcomes.

The sixteenth (**using automated reasoning**) and seventeenth (**using artificial intelligence**) have both been greatly impactful to this project, and will remain a big part of its future.

Acknowledgements

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)

 and

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)

 and

- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)
   and
- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers
- Hoskinson Center for Formal Mathematics

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)
   and
- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers
- Hoskinson Center for Formal Mathematics
- Institute for Computer-Aided Reasoning in Mathematics

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)
   and
- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers
- Hoskinson Center for Formal Mathematics
- Institute for Computer-Aided Reasoning in Mathematics
- G-Research

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)
   and
- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers
- Hoskinson Center for Formal Mathematics
- Institute for Computer-Aided Reasoning in Mathematics
- G-Research

Thank you very much! 감사합니다!

Questions**?**

[1] J. Avigad.
Mathematics and the formal turn.
arXiv preprint arXiv:2311.00007, 2023.

[2] H. Cohn and N. Elkies.
New Upper Bounds on Sphere Packings I.
Annals of Mathematics, 157(2):689–714, 2003.

[3] S. Hariharan.
Viazovska's magic function in dimension 8: An attempt at formalisation.
MSci Thesis, Imperial College London, 2025.
Supervised by Bhavik Mehta.

[4] S. Lee.
Algebraic proof of modular form inequalities for optimal sphere packings.
arXiv preprint arXiv:2406.14659, 2024.

[5] M. S. Viazovska.
The sphere packing problem in dimension 8.
Annals of Mathematics, 185(3):991–1015, 2017.