The Carleson Project: Collaboration using Formalization

Floris van Doorn

University of Bonn

7 April 2025

https://florisvandoorn.com/carleson/

Overview

Overview:

- Overview of Lean and Mathlib;
- Statement of Carleson's theorem;
- Organization of the formalization.

Formalization

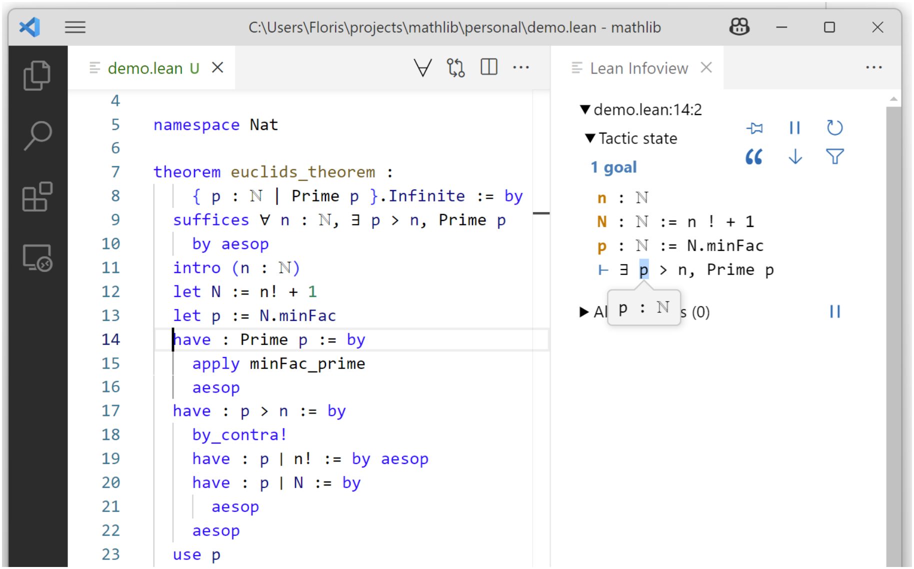
Lean is a programming language and interactive theorem prover.

It is open source, and under active development since 2013.

In it, you can write mathematical definitions, theorem statements and detailed proofs.

These proofs are checked by Lean, down to the axioms of mathematics. This is called formalization.

Lean


C:\Users\Floris\projects\mathlib\personal\demo.lean - mathlib
demo.lean U
Lean Infoview

```
namespace Nat

theorem euclids_theorem :
    { p : ℕ | Prime p }.Infinite := by
  suffices ∀ n : ℕ, ∃ p > n, Prime p
    by aesop
  intro (n : ℕ)
  let N := n! + 1
  let p := N.minFac
  have : Prime p := by
    apply minFac_prime
    aesop
  have : p > n := by
    by_contra!
    have : p ∣ n! := by aesop
    have : p ∣ N := by
      aesop
    aesop
  use p
```

```
▼demo.lean:14:2
 ▼Tactic state
  1 goal
  n : ℕ
  N : ℕ := n ! + 1
  p : ℕ := N.minFac
  ⊢ ∃ p > n, Prime p

▶ Al p : ℕ s (0)
```

Lean's mathematical library

Lean has a mathematical library `Mathlib` with results from many fields in mathematics:
algebra, analysis, geometry, probability theory, combinatorics, logic, topology, category theory, ...

It is large: `Mathlib` has ~1.8 million lines of code, with thousands of definitions and theorems written by over 550 contributors.

It is actively developed: There are more than ~200 contributions every week, reviewed by the 28 maintainers and 22 reviewers.

I have worked with Lean since 2014 and am a maintainer of `Mathlib` since 2019.

Exciting projects in Lean

- Definition of a perfectoid space (Buzzard, Commelin, Massot)
- Independence of the continuum hypothesis (FvD, Jesse Han)
- Liquid Tensor Experiment (led by Commelin and Topaz)
- Polynomial Freiman–Rusza conjecture (led by Terrence Tao)
- Sphere eversion project (FvD, Massot, Nash)
- AlphaProof used Lean to solve IMO problems (Google Deepmind)
- Fermat's Last Theorem project (led by Kevin Buzzard)
- Equational theories (led by Terrence Tao)
- Carleson project (led by FvD)

Exciting projects in Lean

- Definition of a perfectoid space (Buzzard, Commelin, Massot)
- Independence of the continuum hypothesis (FvD, Jesse Han)
- Liquid Tensor Experiment (led by Commelin and Topaz)
- Polynomial Freiman–Rusza conjecture (led by Terrence Tao)
- Sphere eversion project (FvD, Massot, Nash)
- AlphaProof used Lean to solve IMO problems (Google Deepmind)
- Fermat's Last Theorem project (led by Kevin Buzzard)
- Equational theories (led by Terrence Tao)
- Carleson project (led by FvD)

Exciting projects in Lean

- Definition of a perfectoid space (Buzzard, Commelin, Massot)
- Independence of the continuum hypothesis (FvD, Jesse Han)
- Liquid Tensor Experiment (led by Commelin and Topaz)
- Polynomial Freiman–Rusza conjecture (led by Terrence Tao)
- Sphere eversion project (FvD, Massot, Nash)
- AlphaProof used Lean to solve IMO problems (Google Deepmind)
- Fermat's Last Theorem project (led by Kevin Buzzard)
- Equational theories (led by Terrence Tao)
- Carleson project (led by FvD)

Exciting projects in Lean

- Definition of a perfectoid space (Buzzard, Commelin, Massot)
- Independence of the continuum hypothesis (FvD, Jesse Han)
- Liquid Tensor Experiment (led by Commelin and Topaz)
- Polynomial Freiman–Rusza conjecture (led by Terrence Tao)
- Sphere eversion project (FvD, Massot, Nash)
- AlphaProof used Lean to solve IMO problems (Google Deepmind)
- Fermat's Last Theorem project (led by Kevin Buzzard)
- Equational theories (led by Terrence Tao)
- Carleson project (led by FvD)

Exciting projects in Lean

- Definition of a perfectoid space (Buzzard, Commelin, Massot)
- Independence of the continuum hypothesis (FvD, Jesse Han)
- Liquid Tensor Experiment (led by Commelin and Topaz)
- Polynomial Freiman–Rusza conjecture (led by Terrence Tao)
- Sphere eversion project (FvD, Massot, Nash)
- AlphaProof used Lean to solve IMO problems (Google Deepmind)
- Fermat's Last Theorem project (led by Kevin Buzzard)
- Equational theories (led by Terrence Tao)
- Carleson project (led by FvD)

Fourier transform: Definition

Carleson's theorem is an important theorem about the Fourier transform of a function with a notoriously difficult proof.

Fourier transform: Definition

Carleson's theorem is an important theorem about the Fourier transform of a function with a notoriously difficult proof.

Definition

Let $f : \mathbb{R} \to \mathbb{C}$ be an integrable function. Then its Fourier transform $\mathcal{F}f : \mathbb{R} \to \mathbb{C}$ is defined as

$$\mathcal{F}f(\xi) := \int_{\mathbb{R}} f(x)e^{-2\pi i\xi x}\,dx.$$

The inverse Fourier transform $\mathcal{F}^{-1}$ is

$$\mathcal{F}^{-1}g(x) := \int_{\mathbb{R}} g(\xi)e^{2\pi i x\xi}\,d\xi.$$

Fourier transform: Intuition

- We think of the Fourier transform as decomposing a function into the basis elements $x \mapsto e^{2\pi i x\xi}$, like splitting a vector into basis components.
- These basis elements are eigenfunctions of the differentiation operator. So (when f is C^1):

$$\mathcal{F}(f')(\xi) = 2\pi i\xi \mathcal{F}f(\xi).$$

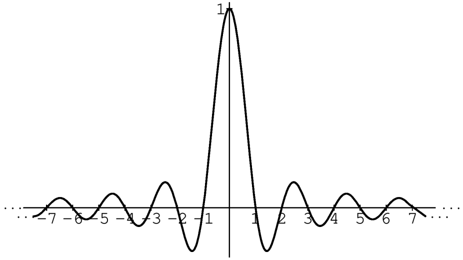
- The decomposition is subtle, since the basis elements $x \mapsto e^{inx}$ themselves are *not* (square) integrable on $\mathbb{R}$.
- When $f \in C^1$, then $\mathcal{F}^{-1}\mathcal{F}f(x) = f(x)$ (Fourier inversion theorem).
- However, if f is merely integrable, $\mathcal{F}f$ need not need to be integrable.

Example: Fourier transform of a box

Example: Let $f := \chi_{[-\frac{1}{2},\frac{1}{2}]}$ be a box function. It has Fourier transform

$$\mathcal{F}f(\xi) = \frac{\sin(\pi\xi)}{\pi\xi}.$$

Note: $\mathcal{F}f$ is not integrable on $\mathbb{R}$.



Improper integrals

If the function is not integrable, but locally integrable, we can define the (inverse) Fourier transform using an improper integral:

$$\mathcal{F}f(\xi) := \lim_{R\to\infty} \int_{[-R,R]} f(x) e^{-2\pi i \xi x}\, dx.$$

Improper integrals

If the function is not integrable, but locally integrable, we can define the (inverse) Fourier transform using an improper integral:

$$\mathcal{F}f(\xi) := \lim_{R\to\infty} \int_{[-R,R]} f(x)e^{-2\pi i\xi x}\,dx.$$

Important: Whether this limit converges depends on the topology you use for this limit:

- Pointwise convergence
- L^p-convergence: $\|f\|_{L^p}^p := \int |f(x)|^p dx$.

Improper integrals

If the function is not integrable, but locally integrable, we can define the (inverse) Fourier transform using an improper integral:

$$\mathcal{F}f(\xi) := \lim_{R\to\infty} \int_{[-R,R]} f(x)e^{-2\pi i\xi x}\,dx.$$

Important: Whether this limit converges depends on the topology you use for this limit:

- Pointwise convergence
- L^p-convergence: $\|f\|_{L^p}^p := \int |f(x)|^p dx$.

If $f \in L^2$ (i.e. $\|f\|_{L^2} < \infty$) then $\mathcal{F}f$ is well-defined using the L^2-norm, and $\mathcal{F}f \in L^2$. In this case, we have $\mathcal{F}^{-1}\mathcal{F}f = f$ w.r.t. to the L^2-norm (Fourier Inversion Theorem).

Carleson's theorem

Theorem (Carleson–Hunt, 1968)

If $f \in L^p$ for some $1 < p \leq 2$. Then for almost every x we have $\mathcal{F}^{-1}\mathcal{F}f(x) = f(x)$.

Carleson proved the case $p = 2$ in 1966.

Carleson's theorem: remarks

- We cannot remove the "almost every" from the statement: even for continuous L^2 functions the limit might diverge for some x.
- There are L^1 functions where the limit defining $\mathcal{F}^{-1}\mathcal{F}f(x)$ diverges for all points x.
- If f is a function in multiple variables, versions of Carleson's theorem also hold. One has to be very careful about the shape of the integration domain that tends to infinity. If the shape is spherical, then this is still an open problem.

Generalized Carleson

- The proof of Carleson's theorem works by showing that the Carleson operator is bounded:

$$Tf(x) := \sup_{n \in \mathbb{Z}} \left| \int_{\mathbb{R}} f(y) \frac{1}{x-y} e^{iny} dy \right|$$

- We formalize a generalization of the Carleson's theorem, which holds when the domain of the function is an arbitrary doubling metric measure space.
- This was proven in 2023 by Lars Becker, Asgar Jamneshan, Rajula Srivastava and Christoph Thiele.
- I joined in late 2023 to start formalizing the 30-page proof.

Blueprint

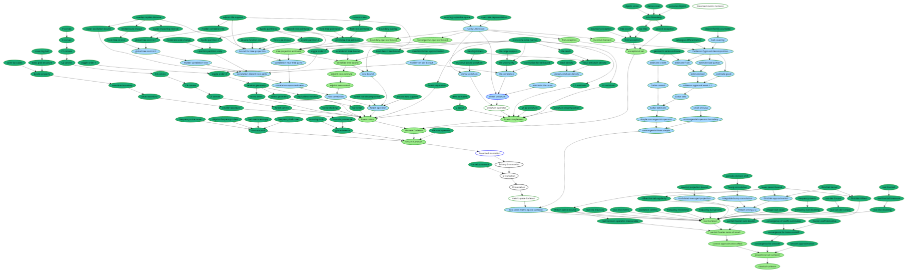
After a false start, the harmonic analysis group wrote a blueprint for the proof of 120 pages, plus 30 pages to prove classical Carleson's theorem as a corollary.

This allows non-experts to take a single lemma and formalize it.

The blueprint has 11 sections.

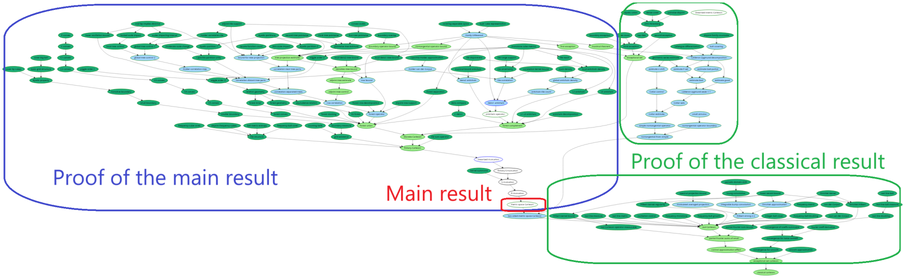
- Section 1: statement of the generalized (metric) Carleson's theorem;
- Section 2: statement of 6 propositions used in the proof;
- Section 3: proof of metric Carleson from the propositions;
- Sections 4-9: each section proves one of the 6 propositions;
- Sections 10-11: proof of the classical Carleson theorem.

Dependency Graph



Currently 138 out of 179 lemmas/theorems formalized

Dependency Graph



Currently 138 out of 179 lemmas/theorems formalized

Formalization

In June 2024 I publicly launched the formalization, and asked volunteers to help.

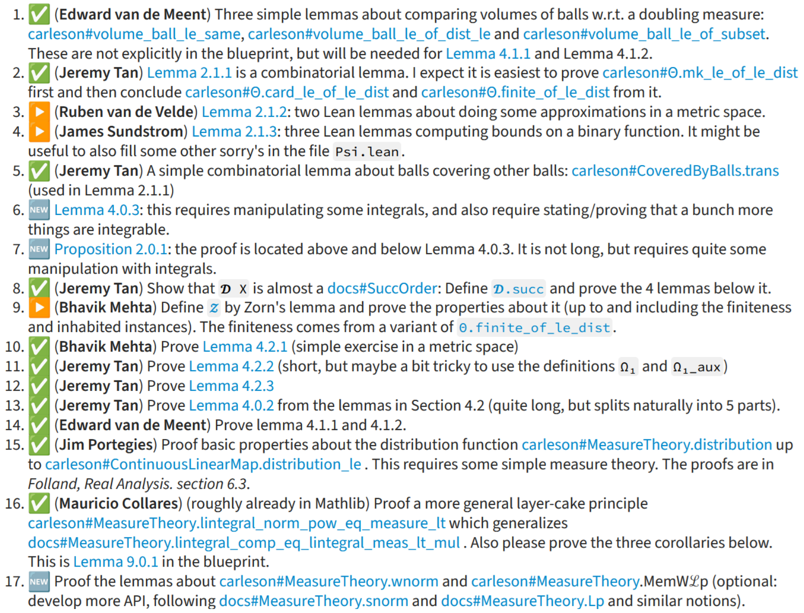
Typically I state the lemmas and definitions in Lean, and then contributors formalized the proof, following the blueprint.

Most contributors do not have a background in harmonic analysis.

The detailed blueprint allows contributors that are not familiar with harmonic analysis to formalize the proofs.

The formalization is j.w.w. María Inés de Frutos-Fernández, Leo Diedering, Sébastien Gouëzel, Evgenia Karunus, Edward van de Meent, Pietro Monticone, Jim Portegies, Michael Rothgang, James Sundstrom, Jeremy Tan, and others.

Collaboration with Lean



1. ✅ (**Edward van de Meent**) Three simple lemmas about comparing volumes of balls w.r.t. a doubling measure: carleson#volume_ball_le_same, carleson#volume_ball_le_of_dist_le and carleson#volume_ball_le_of_subset. These are not explicitly in the blueprint, but will be needed for Lemma 4.1.1 and Lemma 4.1.2.
2. ✅ (**Jeremy Tan**) Lemma 2.1.1 is a combinatorial lemma. I expect it is easiest to prove carleson#Θ.mk_le_of_le_dist first and then conclude carleson#Θ.card_le_of_le_dist and carleson#Θ.finite_of_le_dist from it.
3. ▶️ (**Ruben van de Velde**) Lemma 2.1.2: two Lean lemmas about doing some approximations in a metric space.
4. ▶️ (**James Sundstrom**) Lemma 2.1.3: three Lean lemmas computing bounds on a binary function. It might be useful to also fill some other sorry's in the file `Psi.lean`.
5. ✅ (**Jeremy Tan**) A simple combinatorial lemma about balls covering other balls: carleson#CoveredByBalls.trans (used in Lemma 2.1.1)
6. 🆕 Lemma 4.0.3: this requires manipulating some integrals, and also require stating/proving that a bunch more things are integrable.
7. 🆕 Proposition 2.0.1: the proof is located above and below Lemma 4.0.3. It is not long, but requires quite some manipulation with integrals.
8. ✅ (**Jeremy Tan**) Show that `𝓓 X` is almost a docs#SuccOrder: Define `𝓓.succ` and prove the 4 lemmas below it.
9. ▶️ (**Bhavik Mehta**) Define `𝓩` by Zorn's lemma and prove the properties about it (up to and including the finiteness and inhabited instances). The finiteness comes from a variant of `Θ.finite_of_le_dist`.
10. ✅ (**Bhavik Mehta**) Prove Lemma 4.2.1 (simple exercise in a metric space)
11. ✅ (**Jeremy Tan**) Prove Lemma 4.2.2 (short, but maybe a bit tricky to use the definitions `Ω₁` and `Ω₁_aux`)
12. ✅ (**Jeremy Tan**) Prove Lemma 4.2.3
13. ✅ (**Jeremy Tan**) Prove Lemma 4.0.2 from the lemmas in Section 4.2 (quite long, but splits naturally into 5 parts).
14. ✅ (**Edward van de Meent**) Prove lemma 4.1.1 and 4.1.2.
15. ✅ (**Jim Portegies**) Proof basic properties about the distribution function carleson#MeasureTheory.distribution up to carleson#ContinuousLinearMap.distribution_le . This requires some simple measure theory. The proofs are in *Folland, Real Analysis. section 6.3.*
16. ✅ (**Mauricio Collares**) (roughly already in Mathlib) Proof a more general layer-cake principle carleson#MeasureTheory.lintegral_norm_pow_eq_measure_lt which generalizes docs#MeasureTheory.lintegral_comp_eq_lintegral_meas_lt_mul . Also please prove the three corollaries below. This is Lemma 9.0.1 in the blueprint.
17. 🆕 Proof the lemmas about carleson#MeasureTheory.wnorm and carleson#MeasureTheory.MemWℒp (optional: develop more API, following docs#MeasureTheory.snorm and docs#MeasureTheory.Lp and similar notions).

Collaboration with Lean

Lean enables big cooperations:

- Individual contributors work on their own part of the proof.
- Lean will ensure that the different parts fit together.

Collaboration with Lean

Lean enables big cooperations:

- Individual contributors work on their own part of the proof.
- Lean will ensure that the different parts fit together.

- Lean allows for safe refactoring: When a definition or theorem is reformulated, Lean will inform you of all the places that have to be adapted.

Carleson and Mathlib

The Carleson Project extensively uses `Mathlib`, using properties of integration, metric spaces, measure theory, topology and much more.

The proof is a mix of preliminary results and results that are specific to this proof.

We prove the preliminary results in high generality, so that they can be upstreamed to the `Mathlib`, and reused for other proofs.

Examples are the Marcinkiewicz Interpolation Theorem and the Hardy–Littlewood Maximal Principle.

The specific results aren't necessarily done in the proper generality, and their proofs do not follow Mathlib-standards.

Design Decisions

- In analysis/measure theory, three types are very important:
  - The reals $\mathbb{R}$;
  - The nonnegative reals $[0, \infty) \subseteq \mathbb{R}$;
  - The extended nonnegative reals $[0, \infty]$.
- You have to choose which of these you want to use when stating a result or definition.
- There are canonical maps between them, not all of which behave nicely.
- It is annoying to reason about these maps and cancel them in proofs.

Design Decisions

- At first, in the Carleson project, we decided to just use $\mathbb{R}$ everywhere.
    - All measures, integrals and suprema we work with should be finite.
- This turned out to be the wrong decision.

Design Decisions

- At first, in the Carleson project, we decided to just use $\mathbb{R}$ everywhere.
  - All measures, integrals and suprema we work with should be finite.
- This turned out to be the wrong decision.
  - Even when a supremum (integral/measure) is provably finite, it is often still easier to work with the version that lands in $[0, \infty]$.
  - `Mathlib` likes to work with the operations in $[0, \infty]$.

Design Decisions

- At first, in the Carleson project, we decided to just use $\mathbb{R}$ everywhere.
  - All measures, integrals and suprema we work with should be finite.
- This turned out to be the wrong decision.
  - Even when a supremum (integral/measure) is provably finite, it is often still easier to work with the version that lands in $[0, \infty]$.
  - `Mathlib` likes to work with the operations in $[0, \infty]$.
- We then transitioned into using $[0, \infty]$ extensively, and also generalized some definitions in `Mathlib` to also work on $[0, \infty]$ (e.g. the L^p-norm).

Carleson's theorem in Lean

```
theorem classical_carleson {c : ℝ} (hc : c > 0)
    {f : ℝ → ℂ} (h1 : Continuous f) (h2 : f.Periodic c) :
    ∀ᵐ x : ℝ, Tendsto (partialFourierSum f · x) atTop (𝓝 (f x))
```

Carleson's theorem in Lean

```
theorem metric_carleson [CompatibleFunctions ℝ X (2 ^ a)]
    [IsCancellative X (2 ^ a)] [IsOneSidedKernel a K]
    (ha : 4 ≤ a) (hq : q ∈ Ioc 1 2) (hqq' : q.IsConjExponent q')
    (hF : MeasurableSet F) (hG : MeasurableSet G)
    (hT : HasBoundedStrongType (ANCZOperator K) 2 2 volume volume (C_Ts a))
    (f : X → ℂ) (hf : ∀ x, ‖f x‖ ≤ F.indicator 1 x) :
    ∫⁻ x in G, CarlesonOperator K f x ≤
    ENNReal.ofReal (C1_2 a q) * (volume G) ^ q'⁻¹ * (volume F) ^ q⁻¹ := by
  sorry
```

In conclusion

- Lean is a language with a lot of exciting developments.
- It is feasible to formalize current research in harmonic analysis.
- Large formalization projects can be efficiently divided into small parts, and with a detailed blueprint many people can efficiently contribute.

Thank you for listening

`https://florisvandoorn.com/carleson/`

Design Decisions

The L^p-norm is defined for functions that take values in a Banach space.

In harmonic analysis, we often work with suprema of multiple (nonnegative) functions, e.g. the Hardy–Littlewood maximal function:

$$Mf(x) \coloneqq \sup_{B \ni x} \frac{1}{|B|} \int_B |f|.$$

These functions can equal ∞ on some values (usually only on a set of measure 0), so their codomain is $[0, \infty]$, which is not a Banach space.

To conveniently deal formalize results about these functions, we extend the definition of L^p-norm to also include functions into $[0, \infty]$. This requires axiomatizing the common structure of $[0, \infty]$ and Banach spaces.