Progress report on the Carleson Project

Floris van Doorn

University of Bonn

16 Jan 2025

https://florisvandoorn.com/carleson/

Outline

- Organization of the formalization
- Mathematical statement of the Carleson theorem
- Design decisions

Timeline

- 1966: Carleson's theorem proven by Lennart Carleson
- 2023: Bonn harmonic analysis group proves a generalized version

Timeline

- 1966: Carleson's theorem proven by Lennart Carleson
- 2023: Bonn harmonic analysis group proves a generalized version
- October 2023: discussions start to formalize this result
- November 2023: false start

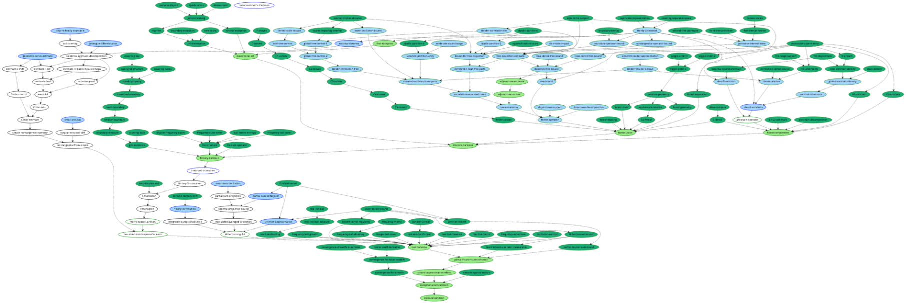
Timeline

- 1966: Carleson's theorem proven by Lennart Carleson
- 2023: Bonn harmonic analysis group proves a generalized version
- October 2023: discussions start to formalize this result
- November 2023: false start
- February-May 2024: write blueprint
- June 2024: public launch of formalization

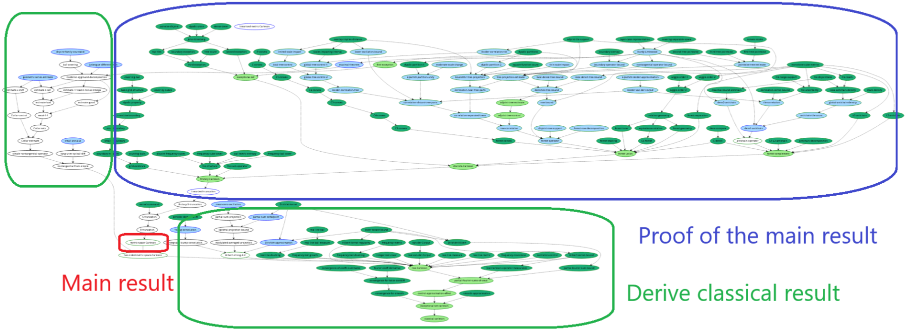
Timeline

- 1966: Carleson's theorem proven by Lennart Carleson
- 2023: Bonn harmonic analysis group proves a generalized version
- October 2023: discussions start to formalize this result
- November 2023: false start
- February-May 2024: write blueprint
- June 2024: public launch of formalization
- September 2024: "50%" done
- Today: progress still going well, although significantly slower after September (109/183 lemmas).

Carleson's theorem in Lean



Carleson's theorem in Lean



Collaboration with Lean

In June I asked volunteers to help.

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

Most contributors did not have a background in Fourier analysis.

Blueprint is j.w.w. Lars Becker, Asgar Jamneshan, Rajula Srivastava, Christoph Thiele.

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 Ω_1 and Ω_1_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).

Blueprint

The original proof was about 30 pages, but that became 120 pages when writing the proofs out in detail, plus 30 pages to prove classical Carleson's theorem as a corollary.

It has 11 chapters:

- 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.

Experiences

- With a detailed blueprint it is possible for experienced Lean users to contribute proofs of many lemmas, even if they don't know the mathematics.
- However: it's harder to reformulate/generalize results

Experiences

- With a detailed blueprint it is possible for experienced Lean users to contribute proofs of many lemmas, even if they don't know the mathematics.
- However: it's harder to reformulate/generalize results

- For the main part of the proof, we do *not* follow Mathlib-standards.
- Many preliminaries are done in proper generality, and will be upstreamed: operators with weak/strong type, real interpolation theorem, Hardy–Littlewood maximal principle and things like Hölder–van der Corput, Hilbert kernels, ...

Fourier transform: Background

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

Fourier transform: Background

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

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

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 $f \in L^2$ 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.

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.

Carleson's operator

The Carleson operator is a sublinear operator that is roughly defined as

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

for $f : \mathbb{R} \to \mathbb{C}$.

Carleson's operator

The Carleson operator is a sublinear operator that is roughly defined as

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

for $f : \mathbb{R} \to \mathbb{C}$.

The vast majority of the work in the proof of Carleson's theorem is to show that this operator is bounded from L^2 to itself, i.e. $\|Tf\|_{L^2} \le C\|f\|_{L^2}$ for some constant C.

Generalizing Carleson's operator

For $f : \mathbb{R} \to \mathbb{C}$

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

Generalizing Carleson's operator

For $f : \mathbb{R} \to \mathbb{C}$

$$Tf(x) := \sup_{n \in \mathbb{Z}} \sup_{0 < R_1 < R_2} \left| \int_{R_1 < |x-y| < R_2} f(y) \frac{1}{x-y} e^{iny} dy \right| .$$

Generalizing Carleson's operator

For $f : \mathbb{R} \to \mathbb{C}$

$$Tf(x) := \sup_{n\in\mathbb{Z}} \sup_{0<R_1<R_2} \left| \int_{R_1<|x-y|<R_2} f(y) K(x,y) e^{iny} dy \right| .$$

Here $K : \mathbb{R} \times \mathbb{R} \to \mathbb{C}$ is a Calderón-Zygmund kernel.

Generalizing Carleson's operator

For $f : \mathbb{R} \to \mathbb{C}$

$$Tf(x) := \sup_{\theta \in \Theta} \sup_{0 < R_1 < R_2} \left| \int_{R_1 < |x-y| < R_2} f(y) K(x,y) e^{i\theta(y)} dy \right| .$$

Here $K : \mathbb{R} \times \mathbb{R} \to \mathbb{C}$ is a Calderón-Zygmund kernel.
Where Θ is a collection of compatible functions $\mathbb{R} \to \mathbb{R}$.

Generalizing Carleson's operator

For $f : X \to \mathbb{C}$

$$Tf(x) := \sup_{\theta \in \Theta} \sup_{0 < R_1 < R_2} \left| \int_{R_1 < d(x,y) < R_2} f(y) K(x,y) e^{i\theta(y)} d\mu(y) \right| .$$

Here $K : X \times X \to \mathbb{C}$ is a Calderón-Zygmund kernel.
Where Θ is a collection of compatible functions $X \to \mathbb{R}$.
X is a doubling metric measure space: a metric space with a Borel measure μ satisfying for some $a \geq 1$

$$\mu(B(x, 2r)) \leq 2^a \mu(B(x, r)) .$$

Generalizing Carleson's operator

For $f : X \to \mathbb{C}$

$$Tf(x) := \sup_{\theta \in \Theta} \sup_{0 < R_1 < R_2} \left| \int_{R_1 < d(x,y) < R_2} f(y) K(x,y) e^{i\theta(y)} d\mu(y) \right| .$$

Here $K : X \times X \to \mathbb{C}$ is a Calderón-Zygmund kernel.
Where Θ is a collection of compatible functions $X \to \mathbb{R}$.
X is a doubling metric measure space: a metric space with a Borel measure μ satisfying for some $a \geq 1$

$$\mu(B(x, 2r)) \leq 2^a \mu(B(x, r)) .$$

In this generality, the generalized Carleson operator is not guaranteed to be bounded, it depends on the boundedness of another simpler operator.

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
```

Design decisions: L^p-spaces

- In analysis, you often look at L^p-spaces: integrable functions that are quotiented by almost everywhere equality.
- Basically everything we do respects a.e. equality.
- We often consider (sub)linear operations $L^p(X) \to L^q(X)$.

Design decisions: L^p-spaces

- In analysis, you often look at L^p-spaces: integrable functions that are quotiented by almost everywhere equality.
- Basically everything we do respects a.e. equality.
- We often consider (sub)linear operations $L^p(X) \to L^q(X)$.
- However, it is painful to work with these quotients in Lean, e.g.

$$(f + g)(x) = f(x) + g(x)$$

will only hold for almost every x.

Design decisions: L^p-spaces

- In analysis, you often look at L^p-spaces: integrable functions that are quotiented by almost everywhere equality.
- Basically everything we do respects a.e. equality.
- We often consider (sub)linear operations $L^p(X) \to L^q(X)$.
- However, it is painful to work with these quotients in Lean, e.g.

$$(f + g)(x) = f(x) + g(x)$$

  will only hold for almost every x.
- Much nicer: work with actual functions, not with quotients.

Design decisions: `ENNReal`

- In analysis/measure theory, three types are very important:
  - The reals $\mathbb{R}$;
  - The nonnegative reals $\mathbb{R}{\geq}0$, defined as `{x : ℝ // 0 ≤ x}`;
  - The extended nonnegative reals $\mathbb{R}{\geq}0\infty$, defined as `WithTop ℝ≥0`.
- In many proofs, you will use multiple of these types, and use the casts/canonical maps between them, not all of which behave nicely.
- It is a major pain to reason about these casts/cancel them in proofs.

Design decisions: `ENNReal`

- Proposal: in the Carleson project, just use $\mathbb{R}$ everywhere.
  - All measures, integrals and suprema we work with should be finite.

Design decisions: `ENNReal`

- Proposal: in the Carleson project, just use $\mathbb{R}$ everywhere.
  - All measures, integrals and suprema we work with should be finite.
- This turned out to make the problem worse.
  - Even when a supremum (integral/measure) is provably finite, it is often still easier to work with the version that lands in $\mathbb{R}_{\geq 0}\infty$.
  - Mathlib likes to work with the operations in $\mathbb{R}_{\geq 0}\infty$.
  - Some operations were infinite in some cases.

Design decisions: `ENorm`

- Let $f : \mathbb{R}^d \to \mathbb{C}$, the Hardy-Littlewood maximal function of f is defined to be $Mf : \mathbb{R} \to \mathbb{R}$

$$Mf(x) = \sup_{r>0} \frac{1}{|B(x,r)|} \int_{B(x,r)} |f(y)|\, dy.$$

Design decisions: `ENorm`

- Let $f : \mathbb{R}^d \to \mathbb{C}$, the Hardy-Littlewood maximal function of f is defined to be $Mf : \mathbb{R} \to \mathbb{R}{\geq}0\infty$

$$Mf(x) = \sup_{r>0} \frac{1}{|B(x,r)|} \int_{B(x,r)} |f(y)|\, dy.$$

- If f is integrable, then Mf is (weak) L^1, and hence almost everywhere finite.

Design decisions: `ENorm`

- Let $f : \mathbb{R}^d \to \mathbb{C}$, the Hardy-Littlewood maximal function of f is defined to be $Mf : \mathbb{R} \to \mathbb{R}{\geq}0\infty$

$$Mf(x) = \sup_{r>0} \frac{1}{|B(x,r)|} \int_{B(x,r)} |f(y)|\, dy.$$

- If f is integrable, then Mf is (weak) L^1, and hence almost everywhere finite.
- Before: $f \in L^p$ was only defined in Lean for $f : X \to E$ where E was a normed vector space.

Design decisions: `ENorm`

- Let $f : \mathbb{R}^d \to \mathbb{C}$, the Hardy-Littlewood maximal function of f is defined to be $Mf : \mathbb{R} \to \mathbb{R}{\geq}0\infty$

$$Mf(x) = \sup_{r>0} \frac{1}{|B(x,r)|} \int_{B(x,r)} |f(y)|\, dy.$$

- If f is integrable, then Mf is (weak) L^1, and hence almost everywhere finite.
- Before: $f \in L^p$ was only defined in Lean for $f : X \to E$ where E was a normed vector space.
- This means that we could only *state* in Lean that $x \mapsto (Mf(x))$.toReal is L^1.
- So we couldn't conclude from this that it is almost everywhere finite.

Design decisions: ENorm

- Solution: introduce a notation class

```
class ENorm (E : Type*) where
  enorm : E → ℝ≥0∞
```

- Normed spaces and ℝ≥0∞ are both instances of his this class.
- Definitions like `Integrable` and `MemLp` can be generalized to functions where the codomain has a `ENorm` class.

Design decisions: ENorm

- Solution: introduce a notation class

```
class ENorm (E : Type*) where
  enorm : E → ℝ≥0∞
```

- Normed spaces and ℝ≥0∞ are both instances of his this class.
- Definitions like `Integrable` and `MemLp` can be generalized to functions where the codomain has a `ENorm` class.
- To generalize result to `ENorm`, we need some laws, potentially this:

```
class ENormedSpace (E : Type*) extends
  ENorm E, AddCommMonoid E, Module ℝ≥0 E where
  continuous_enorm : Continuous enorm
  enorm_eq_zero : ∀ x : E, ‖x‖ₑ = 0 ↔ x = 0
  enorm_add_le : ∀ x y : E, ‖x + y‖ₑ ≤ ‖x‖ₑ + ‖y‖ₑ
  enorm_smul : ∀ (c : ℝ≥0) (x : E), ‖c • x‖ₑ = c • ‖x‖ₑ
```

In conclusion

- 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 non-experts can contribute to the (easier) tasks.
- We've identified various useful design decisions for analysis.

Thank you for listening

http://florisvandoorn.com/carleson/