FORMALIZING CARLESON'S THEOREM IN LEAN

LARS BECKER, MARÍA INÉS DE FRUTOS-FERNÁNDEZ, LEO DIEDERING,
FLORIS VAN DOORN, SÉBASTIEN GOUËZEL, EVGENIA KARUNUS, EDWARD VAN DE MEENT,
PIETRO MONTICONE, JASPER MULDER-SOHN, JIM PORTEGIES, JORIS ROOS,
MICHAEL ROTHGANG, JAMES SUNDSTROM, AND JEREMY TAN

Abstract. We present the formalization of Carleson's theorem in the proof assistant *Lean*. This paper describes the mathematical content, organization of the project, the blueprint, and the main design decisions behind the formalization. It is the result of a large collaborative effort, written and developed in public.

Contents

Date: July 16, 2026.

1. Introduction

1.1. **Brief history and significance of Carleson's theorem.** One of the oldest questions in Fourier analysis consists of understanding the nature of convergence of Fourier series for different classes of functions (see Definition 2.1). There are two main kinds of convergence that are commonly considered. First, one can ask whether the partial Fourier sums S_N converge to the function f in a suitable function space with a given topology. There are several well-known answers to this question. For instance, convergence in the Hilbert space sense for f in $L^2(0, 2\pi)$ was established in the first decade of the twentieth century as a consequence of the rapid development of Lebesgue integration theory, and convergence in the sense of distributions for general distributional f was discovered a few decades after Lebesgue integration. For some other natural spaces, such as $L^1(0, 2\pi)$, there is no guarantee of convergence in the norm of that space, even if f is in the space.

A second question is whether the sequence $\{S_N f(x)\}_{\{N\in\mathbb{N}\}}$ converges pointwise to $f(x)$ for almost every x. This turned out to be a significantly harder problem. Luzin [Luz13] conjectured that this convergence would hold for functions in the $L^2(0, 2\pi)$ space. In 1923, Kolmogorov found an example [Kol23] of an L^1 function whose Fourier series diverges at almost every point, which seemed to suggest that the conjecture might be false. However, when in the 1960s Carleson tried to construct a counterexample to Luzin's conjecture, he realized that a potential counterexample would have to satisfy so many properties that they would in fact be contradictory. He was therefore able to conclude that such a counterexample could not exist, hence proving Luzin's conjecture [Car66].

Carleson's result was soon after generalized by Hunt [Hun68] to the case of L^p functions with $p > 1$ and for functions in $L^1 \log(L)^2$. Billard [Bil67] adapted Carleson's arguments to prove almost everywhere convergence of Walsh–Fourier series for functions in L^2.

Fefferman gave an alternative proof of Carleson's theorem [Fef73] via an a priori bound for a certain maximal operator in harmonic analysis, known as the Carleson operator. Lacey and Thiele pointed out a duality between the approaches by Carleson and Fefferman and presented a more symmetric and self-dual proof [LT00].

In recent years, the impact of Carleson's theorem has increased thanks to its connections to areas of mathematics and physics including ergodic theory and scattering theory, and various strengthenings and variants of Fefferman's estimates for Carleson's operator have appeared; see [Bec+24c, §1]. For more details about Carleson's theorem and its generalizations, we refer the reader to the surveys in [Lac04] and [Dem15].

In this article, we focus on a generalization of Carleson's theorem to the setting of doubling metric measure spaces, proven in [Bec+24a].

1.2. **Overview of the formalization project.** The goal of the Carleson project was to formalize the proof of the metric space Carleson theorem [ref], proven in [Bec+24b], as well as the proof of the classical Carleson theorem [ref] as a corollary of this more general result.

The original reference [Bec+24b] was written as a traditional paper for experts in harmonic analysis, but its formalization had from the beginning been envisioned as a large scale collaborative project. This posed a problem, since there are not many experts in formalization that are also experts in harmonic analysis, so most of the potential contributors to the formalization would not have the required expertise to fill in the gaps that are considered routine arguments in a harmonic analysis paper.

To address this challenge, before the formalization started, the authors of [Bec+24b] wrote a much more detailed document [Bec+24c] intended as a blueprint for the formalization, which was modified as needed while the formalization process was taking place (see §4).

The Carleson formalization project was publicly announced and opened to contributors in June 2024, and it was completed in July 2025. The core authors of the formalization are the 14 authors of this paper, and smaller contributions were made by the 14 additional people listed in the acknowledgements section. This makes the Carleson project one of the largest scale formalization efforts to date.

The formalization was written in Lean 4, making extensive use of Lean's mathematical library, `Mathlib`. The code was developed in a public GitHub repository[1], with an associated website[2] that also linked to HTML and PDF versions of the blueprint. Coordination among project collaborators took place using the Lean community Zulip chat[3]. For more details, see §3.

The project comprises around 35.6k lines of Lean code (not including spaces or comments), of which around 9.6k are intended to be upstreamed to the `Mathlib` library, a process that is still ongoing. Pull requests into `Mathlib` coming from this project are marked with the `carleson` tag; at the time of writing this article, this includes 90 merged pull requests and 11 more under review.

Throughout the article, we will introduce Lean code excerpts from the formalization and use the clickable symbol ↗ to link to the corresponding code in our repository or in Mathlib, as appropriate. Some of the code excerpts have been slightly edited for readability.

1.3. **Related work.**

- Other large-scale formalizations in Lean.
- Other harmonic analysis formalizations.

Acknowledgement. The authors are grateful to Asgar Jamneshan, Rajula Srivastava and Christoph Thiele for writing the mathematical paper [cite] and the blueprint [cite] on which this formalization is based. We furthermore acknowledge contributions in the form of small formalization additions, pointing out corrections to the blueprint, or supplying ideas to the Lean efforts by the following people: Michel Alexis, Bolton Bailey, Julian Berman, Joachim Breitner, Martin Dvořák, Georges Gonthier, Aaron Hill, Austin Letson, Bhavik Mehta, Eric Paul, Clara Torres, Dennis Tsar, Andrew Yang, and Ruben van de Velde.

L.B., M.I.d.F.F., L.D., F.v.D., M.R. were funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2047/1 – 390685813. L.B. was also supported by SFB 1060. J.R. was supported in part by NSF grant DMS-2154835 and a HIM fellowship for the Fall 2024 trimester program in Bonn.

2. The Metric Space Carleson theorem

2.1. **Mathematical background.**

The basic object of study in Carleson's theorem is the Fourier series of a periodic function on the real numbers. Given a 2π-periodic function $f : \mathbb{R} \to \mathbb{C}$ and $n \in \mathbb{Z}$, define the *nth Fourier coefficient* of f as

$$c_n := \widehat{f}_n := \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-inx}\, dx. \tag{1}$$

[1] https://github.com/fpvandoorn/carleson

[2] https://florisvandoorn.com/carleson/

[3] https://leanprover.zulipchat.com

The Lean version `fourierCoeffOn` uses `Mathlib`'s formalization of the Bochner integral and thus is well-defined if $f \in L^1(0, 2\pi)$.

Definition 2.1. The *N-th partial Fourier sum* of f for $x \in \mathbb{R}$ is defined as

$$S_N f(x) \coloneqq \sum_{n=-N}^{N} c_n e^{inx}. \tag{2}$$

The *Fourier series* of f is then given by the formal trigonometric series $\sum_{n=-\infty}^{\infty} c_n e^{inx}$, which may or may not converge for a given $x \in \mathbb{R}$.

The metric space Carleson theorem in [Bec+24c] is a restricted weak type estimate for a generalized Carleson operator. One of the generalizations here is that the domain of the functions operated on can be any *doubling metric measure space* (X, ρ, μ, a).

Definition 2.2. A *doubling metric measure space* (X, ρ, μ, a) is a complete and locally compact metric space (X, ρ) equipped with a non-zero locally finite Borel measure μ that satisfies the following doubling condition for $a \in \mathbb{N}$. For all $x \in X$ and all $R > 0$ we have

$$\mu(B(x, 2R)) \leq 2^a \mu(B(x, R)), \tag{3}$$

where $B(x, R) \coloneqq \{y \in X : \rho(x, y) < R\}$ denotes the open ball of radius R centered at x (see `DoublingMeasure`).

The proof of the metric space Carleson theorem relies on several tools from real analysis and part of the formalization is a proof of the Marcinkiewicz real interpolation theorem (see 4.2.4). Statement and proof require the following definitions for which we could rely on `Mathlib`'s measure theory (see e.g. [Doo21] for an overview). For a function f on a (not necessarily doubling) measure space (X, μ) taking values in some space E endowed with an extended norm (see 5.4), we define its weak L^p norm to be

$$\|f\|_{L^{p,\infty}} \coloneqq \sup_{t>0} t \cdot \mu(\{x \in X : t < |f(x)|\})^{\frac{1}{p}} \tag{4}$$

if $0 < p < \infty$, and $\|f\|_{L^{\infty,\infty}} \coloneqq \|f\|_{L^\infty}$. The function f is said to be in weak L^p (`MemWLp`) if f is μ-a.e. strongly measurable with respect to a fixed topology on E (see e.g. [Doo21; Gou22] for motivation of the different notions of measurability in mathlib) and $\|f\|_{L^{p,\infty}} < \infty$. An important consequence is that if f is in weak L^p, then $|f|$ is finite almost everywhere (see `MemWLp.ae_ne_top`).

Given two measure spaces (X, μ) and (Y, ν), and two topological spaces E and F endowed with extended norms, we say that an operator $T : (X \to E) \to (Y \to F)$ is of weak type (p, p') if there exists a constant C such that for any $f \in L^p(X; E)$, we have that Tf is ν-a.e. strongly measurable and

$$\|Tf\|_{L^{p',\infty}} \leq C \cdot \|f\|_{L^p}. \tag{5}$$

Similarly, the operator T is of strong type if there exists a constant C such that for any $f \in L^p(X; E)$, we have that Tf is ν-a.e. strongly measurable and

$$\|Tf\|_{L^{p'}} \leq C \cdot \|f\|_{L^p}. \tag{6}$$

2.2. **Statement of the main results.**

2.2.1. *Classical Carleson.* The namesake of the formalization project is the classical result of Carleson on the pointwise convergence of Fourier series. The specific statement is as follows.

Theorem 2.2.1 (classical Carleson)**.** *Let f be a 2π-periodic complex-valued continuous function on $\mathbb{R}$. Then for almost all $x \in \mathbb{R}$, the limit $\lim_{N\to\infty} S_N f(x)$ exists and is equal to $f(x)$.*

The formalization was approached through a recent result (blabla reference) which encompasses a number of generalizations of Carleson's original theorem. The context for this new result is called a *doubling metric measure space*, accompanied by a *cancellative compatible collection* of real-valued continuous functions. For a *one-sided* respectively *two-sided Calderón–Zygmund kernel* K on this space, one obtains bounds for certain *generalized Carleson operators* involving the collection of functions and the kernel.

The classical Carleson theorem is obtained by choosing a specific Carleson operator, an approach that all proofs of Carleson's theorem share. The specific version that has been formalized is this one:

Theorem 2.2.2 (real Carleson)**.** *Let F, G be Borel subsets of $\mathbb{R}$ with finite measure. Let f be a bounded measurable function on $\mathbb{R}$ with $|f| \le \mathbf{1}_F$. Then*

$$\left|\int_G Tf(x)\,dx\right| \le 2^{474\cdot 4^3}|F|^{\frac{1}{2}}|G|^{\frac{1}{2}}\,, \tag{7}$$

and

$$Tf(x) := \sup_{n\in\mathbb{Z}}\sup_{r>0}\left|\int_{r<|x-y|<1} f(y)\kappa(x-y)e^{iny}\,dy\right|\,, \tag{8}$$

with $\kappa : \mathbb{R} \to \mathbb{C}$ defined as

$$\kappa(x) := \begin{cases} 0 & \text{if } x = 0 \text{ or } |x| \ge 1 \\ \frac{1-|x|}{1-e^{ix}} & \text{if } 0 < |x| < 1. \end{cases} \tag{9}$$

2.2.2. *Preliminary definitions.* Let $a \ge 4$ be a natural number. A collection Θ of real-valued continuous functions on the doubling metric measure space (X, ρ, μ, a) is called *compatible* if there is a point $o \in X$ where all the functions are equal to 0, and if there exists for each ball $B \subset X$ a metric d_B on Θ, such that the following five properties (10), (11), (12), (13), and (14) are satisfied. For every ball $B \subset X$ and any $\vartheta, \theta \in \Theta$,

$$\sup_{x,y\in B} |\vartheta(x) - \vartheta(y) - \theta(x) + \theta(y)| \le d_B(\vartheta, \theta)\,. \tag{10}$$

For any two balls $B_1 = B(x_1, R)$, $B_2 = B(x_2, 2R)$ in X with $x_1 \in B_2$ and any $\vartheta, \theta \in \Theta$,

$$d_{B_2}(\vartheta, \theta) \le 2^a d_{B_1}(\vartheta, \theta). \tag{11}$$

For any two balls B_1, B_2 in X with $B_1 \subset B_2$ and any $\vartheta, \theta \in \Theta$

$$d_{B_1}(\vartheta, \theta) \le d_{B_2}(\vartheta, \theta) \tag{12}$$

and for any two balls $B_1 = B(x_1, R)$, $B_2 = B(x_2, 2^a R)$ with $B_1 \subset B_2$, and $\vartheta, \theta \in \Theta$,

$$2d_{B_1}(\vartheta, \theta) \le d_{B_2}(\vartheta, \theta). \tag{13}$$

For every ball B in X and every d_B-ball $\tilde{B}$ of radius $2R$ in Θ, there is a collection $\mathcal{B}$ of at most 2^a many d_B-balls of radius R covering $\tilde{B}$, that is,

$$\tilde{B} \subset \bigcup \mathcal{B}. \tag{14}$$

Further, a compatible collection Θ is called *cancellative* if for any ball B in X of radius R, any Lipschitz function $\varphi : X \to \mathbb{C}$ supported on B, and any $\vartheta, \theta \in \Theta$ we have

$$|\int_B e(\vartheta(x) - \theta(x))\varphi(x)d\mu(x)| \leq 2^a \mu(B)\|\varphi\|_{\mathrm{Lip}(B)}(1 + d_B(\vartheta, \theta))^{-\frac{1}{a}}, \tag{15}$$

where $\|\cdot\|_{\mathrm{Lip}(B)}$ denotes the inhomogeneous Lipschitz norm on B:

$$\|\varphi\|_{\mathrm{Lip}(B)} = \sup_{x\in B} |\varphi(x)| + R \sup_{x,y\in B, x\neq y} \frac{|\varphi(x) - \varphi(y)|}{\rho(x, y)}.$$

Definition 2.3. A *one-sided Calderón–Zygmund kernel* K on the doubling metric measure space (X, ρ, μ, a) is a measurable function $K : X \times X \to \mathbb{C}$ such that for all $x, y', y \in X$ with $x \neq y$, we have

$$|K(x, y)| \leq \frac{2^{a^3}}{V(x, y)} \tag{16}$$

and if $2\rho(y, y') \leq \rho(x, y)$, then

$$|K(x, y) - K(x, y')| \leq \left(\frac{\rho(y, y')}{\rho(x, y)}\right)^{\frac{1}{a}} \frac{2^{a^3}}{V(x, y)}, \tag{17}$$

where $V(x, y) := \mu(B(x, \rho(x, y)))$.

Definition 2.4. A *two-sided Calderón–Zygmund kernel* is a one-sided Calderón–Zygmund kernel K which additionally satisfies for all $x, x', y \in X$ with $x \neq y$ and $2\rho(x, x') \leq \rho(x, y)$,

$$|K(x, y) - K(x', y)| \leq \left(\frac{\rho(x, x')}{\rho(x, y)}\right)^{\frac{1}{a}} \frac{2^{a^3}}{V(x, y)}. \tag{18}$$

Let $f : X \to \mathbb{C}$ be a bounded, measurable function supported on a set of finite measure. Define the *maximally truncated non-tangential singular integral* T_* associated with K by

$$T_* f(x) := \sup_{R_1<R_2} \sup_{\rho(x,x')<R_1} \left|\int_{R_1<\rho(x',y)<R_2} K(x', y)f(y)\,\mathrm{d}\mu(y)\right|. \tag{19}$$

Additionally, define for $r > 0$, $x \in X$:

$$T_r f(x) := \int_{r\leq\rho(x,y)} K(x, y)f(y)\,d\mu(y) = \int_{X\setminus B(x,r)} K(x, y)f(y)\,d\mu(y). \tag{20}$$

Finally, define the *generalized Carleson operator* T by

$$Tf(x) := \sup_{\vartheta\in\Theta} \sup_{0<R_1<R_2} \left|\int_{R_1<\rho(x,y)<R_2} K(x, y)f(y)e(\vartheta(y))\,\mathrm{d}\mu(y)\right|, \tag{21}$$

where $e(r) = e^{ir}$.

For a Borel function $Q : X \to \Theta$, and $\vartheta \in \Theta$ and $x \in X$ define

$$R_Q(\vartheta, x) = \sup\{r > 0 : d_{B(x,r)}(\vartheta, Q(x)) < 1\} \tag{22}$$

and define further

$$T_Q^\vartheta f(x) := \sup_{R_1<R_2} \sup_{\rho(x,x')<R_1} \left|\int_{R_1<\rho(x',y)<\min\{R_2,R_Q(\vartheta,x')\}} K(x', y)f(y)\,\mathrm{d}\mu(y)\right|. \tag{23}$$

Define further the *linearized generalized Carleson operator* T_Q by

$$T_Q f(x) := \sup_{0<R_1<R_2} \left| \int_{R_1<\rho(x,y)<R_2} K(x,y) f(y) e(Q(x)(y)) \, \mathrm{d}\mu(y) \right| , \tag{24}$$

where again $e(r) = e^{ir}$.

2.2.3. *Metric space Carleson.* We prove three versions of the main result. The first version is the following restricted weak type estimate for T in the range $1 < q \leq 2$, which by interpolation techniques recovers L^q estimates for the open range $1 < q < 2$.

Theorem 2.2.3 (metric space Carleson)**.** *For all integers $a \geq 4$ and real numbers $1 < q \leq 2$ the following holds. Let (X, ρ, μ, a) be a doubling metric measure space. Let Θ be a cancellative compatible collection of functions and let K be a one-sided Calderón–Zygmund kernel on (X, ρ, μ, a). Assume that for every bounded measurable function g on X supported on a set of finite measure we have*

$$\|T_* g\|_2 \leq 2^{a^3} \|g\|_2 , \tag{25}$$

where T_* is defined in (19). Then for all Borel sets F and G in X and all Borel functions $f : X \to \mathbb{C}$ with $|f| \leq \mathbf{1}_F$, we have, with T defined in (21),

$$\left| \int_G T f \, \mathrm{d}\mu \right| \leq \frac{2^{443a^3}}{(q-1)^6} \mu(G)^{1-\frac{1}{q}} \mu(F)^{\frac{1}{q}} . \tag{26}$$

In some applications, such as the Walsh-case of Carleson's theorem, the kernel K naturally depends also on the modulation functions ϑ. The fact that we don't assume Hölder-continuity of the kernel K in the first argument allows us to also capture this situation.

The second version of our main result has a slightly weaker replacement of the assumption (25). It implies the Walsh-case of Carleson's theorem.

Theorem 2.2.4 (linearized metric space Carleson)**.** *For all integers $a \geq 4$ and real numbers $1 < q \leq 2$ the following holds. Let (X, ρ, μ, a) be a doubling metric measure space. Let Θ be a cancellative compatible collection of functions. Let $Q : X \to \Theta$ be a Borel function with finite range. Let K be a one-sided Calderón–Zygmund kernel on (X, ρ, μ, a). Assume that for every $\vartheta \in \Theta$ and every bounded measurable function g on X supported on a set of finite measure we have*

$$\|T_Q^\vartheta g\|_2 \leq 2^{a^3} \|g\|_2 , \tag{27}$$

where T_Q^ϑ is defined in (23). Then for all Borel sets F and G in X and all Borel functions $f : X \to \mathbb{C}$ with $|f| \leq \mathbf{1}_F$, we have, with T_Q defined in (24),

$$\left| \int_G T_Q f \, \mathrm{d}\mu \right| \leq \frac{2^{443a^3}}{(q-1)^6} \mu(G)^{1-\frac{1}{q}} \mu(F)^{\frac{1}{q}} . \tag{28}$$

The third version of the main result involves two-sided Calderón–Zygmund kernels, whose additional regularity allows us to weaken one of the assumptions to the family of operators T_r, which is easier to work with in applications. In particular, the two-sided case can be applied to derive the classical Carleson's theorem.

Theorem 2.2.5 (two-sided metric space Carleson)**.** *For all integers $a \geq 4$ and real numbers $1 < q \leq 2$ the following holds. Let (X, ρ, μ, a) be a doubling metric measure space. Let Θ be a cancellative compatible collection of functions and let K be a two-sided Calderón–Zygmund*

kernel on (X, ρ, μ, a). *Assume that for every bounded measurable function* g *on* X *supported on a set of finite measure and all* $r > 0$ *we have*

$$\|T_r g\|_2 \leq 2^{a^3} \|g\|_2 \,. \tag{29}$$

Then for all Borel sets F *and* G *in* X *and all Borel functions* $f : X \to \mathbb{C}$ *with* $|f| \leq \mathbf{1}_F$, *we have, with* T *defined in* (21),

$$\left| \int_G T f \, \mathrm{d}\mu \right| \leq \frac{2^{474a^3}}{(q-1)^6} \mu(G)^{1-\frac{1}{q}} \mu(F)^{\frac{1}{q}} \,. \tag{30}$$

2.2.4. *Proof roadmap.* In order to highlight choices made in the proof and the formalization it is practical to give a brief roadmap of the proof of the main results stated above. Of these three, the linearized version Theorem 2.2.4 implies the others. It is proved by first reducing it to a finitary version, which is then reduced to a discrete version.

Given the variables and assumptions of Theorem 2.2.4, define

$$D := 2^{100a^2}$$

and let $\psi : \mathbb{R} \to \mathbb{R}$ be the unique compactly supported, piecewise linear, continuous function with corners precisely at $\frac{1}{4D}$, $\frac{1}{2D}$, $\frac{1}{4}$ and $\frac{1}{2}$ which satisfies

$$\sum_{s \in \mathbb{Z}} \psi(D^{-s} x) = 1 \tag{31}$$

for all $x > 0$. Note that the above sum has at most two nonzero terms, and that this function vanishes outside $[\frac{1}{4D}, \frac{1}{2}]$, is constant one on $[\frac{1}{2D}, \frac{1}{4}]$, and is Lipschitz with constant $4D$. With this definition, the finitary Carleson Theorem reads as follows.

Theorem 2.2.6 (finitary Carleson)**.** *Let* $\sigma_1, \sigma_2 \colon X \to \mathbb{Z}$ *be measurable functions with finite range and* $\sigma_1 \leq \sigma_2$. *Let* F, G *be bounded Borel sets in* X. *Then there is a Borel set* G' *in* X *with* $2\mu(G') \leq \mu(G)$ *such that for all Borel functions* $f : X \to \mathbb{C}$ *with* $|f| \leq \mathbf{1}_F$ *we have*

$$\int_{G \setminus G'} \left| \sum_{s=\sigma_1(x)}^{\sigma_2(x)} \int K_s(x,y) f(y) e(Q(x)(y)) \, \mathrm{d}\mu(y) \right| \mathrm{d}\mu(x)$$

$$\leq \frac{2^{442a^3}}{(q-1)^5} \mu(G)^{1-\frac{1}{q}} \mu(F)^{\frac{1}{q}} \tag{32}$$

where for $s \in \mathbb{Z}$, $K_s(x,y) = K(x,y)\psi(D^{-s}\rho(x,y))$.

To reduce this finitary version of the Carleson theorem to the discrete version, we make divisions in both the spatial and frequency domains. The spatial discretization will be achieved by a *grid structure*, after which a subordinate frequency discretization is encoded by a *tile structure*.

A grid structure on X is a triple $(\mathcal{D}, c, s)$ consisting of a finite set $\mathcal{D}$ of so-called dyadic cubes, a function $c : \mathcal{D} \to X$ called the center function, and a surjective function $s : \mathcal{D} \to [-S, S]$ called the scale function, that together satisfy the conditions i.-v. below. Here, a dyadic cube is just a pair (I, k) of a Borel set I in X and an integer $k \in [-S, S]$. Usually we abuse notation and write I for a dyadic cube (I, k).

The conditions that the objects in a grid structure satisfy are as follows.

i. For every dyadic cube I and every $-S \leq k < s(I)$,

$$I \subset \bigcup_{J \in \mathcal{D}: s(J)=k} J\,. \tag{33}$$

ii. For every non-disjoint dyadic cubes I, J with $s(I) \leq s(J)$, we have that

$$I \subset J. \tag{34}$$

iii. The exists a cube $I_0 \in \mathcal{D}$ with scale $s(I_0) = S$ and center $c(I_0) = o$ (the point where all $\vartheta \in \Theta$ vanish), and for all dyadic cubes J it holds that

$$J \subset I_0\,. \tag{35}$$

iv. Every dyadic cube I fits a ball with center $c(I)$ and is approximately of size $D^{s(I)}$ in the sense that

$$B(c(I), \frac{1}{4}D^{s(I)}) \subset I \subset B(c(I), 4D^{s(I)})\,. \tag{36}$$

v. For every dyadic cube I and every t with $tD^{s(I)} \geq D^{-S}$, the boundary of I is small in the sense that

$$\mu(\{x \in I \ : \ \rho(x, X \setminus I) \leq tD^{s(I)}\}) \leq 2t^{\kappa}\mu(I)\,. \tag{37}$$

We now further subdivide in the frequency domain, intuitively partitioning for every dyadic cube the collection of functions Θ into pieces called tiles. More precisely, given a grid structure $(\mathcal{D}, c, s)$, a tile structure for this grid structure is a tuple $(\mathfrak{P}, \mathcal{I}, \Omega, \mathcal{Q}, \mathrm{c}, \mathrm{s})$ where $\mathfrak{P}$ is a finite set with elements called tiles. The function $\Omega : \mathfrak{P} \to \mathcal{P}(\Theta)$ maps every tile to a subset of the collection of functions Θ, and the function $\mathcal{Q} : \mathfrak{P} \to Q(X)$ plays the role of center function for the selection of functions. The association of dyadic cubes to a collection of tiles is encoded in the reverse direction by a surjective function $\mathcal{I} : \mathfrak{P} \to \mathcal{D}$ that merely associates to every tile a dyadic cube and the functions $\mathrm{c} : \mathfrak{P} \to X$ and $\mathrm{s} : \mathfrak{P} \to [-S, S]$ that just yield the center and scale of the given cube. The above functions need to satisfy the following conditions.

i. For every dyadic cube I and every pair of tiles $\mathfrak{p} \neq \mathfrak{q}$ in $\mathcal{I}^{-1}(\{I\})$, the sets $\Omega(\mathfrak{p})$ and $\Omega(\mathfrak{q})$ are disjoint.

ii. For every dyadic cube I, we have that

$$Q(X) \subset \bigcup_{\mathfrak{p} \in \mathcal{I}^{-1}(\{I\})} \Omega(\mathfrak{p}). \tag{38}$$

iii. For all tiles $\mathfrak{p}, \mathfrak{q}$ with $\mathcal{I}(\mathfrak{p}) \subset \mathcal{I}(\mathfrak{q})$ it holds that

$$\Omega(\mathfrak{q}) \subset \Omega(\mathfrak{p}). \tag{39}$$

iv. For every tile $\mathfrak{p}$, the set of functions $\Omega(\mathfrak{p})$ contains a ball with center $\mathcal{Q}(\mathfrak{p})$ and is of size approximately $D^{\mathrm{s}(\mathfrak{p})}$ in the sense that

$$B_{\mathfrak{p}}(\mathcal{Q}(\mathfrak{p}), 0.2) \subset \Omega(\mathfrak{p}) \subset B_{\mathfrak{p}}(\mathcal{Q}(\mathfrak{p}), 1)\,, \tag{40}$$

where

$$B_{\mathfrak{p}}(\vartheta, R) := \{\theta \in \Theta \ : \ d_{\mathfrak{p}}(\vartheta, \theta) < R\}, \tag{41}$$

with

$$d_{\mathfrak{p}} := d_{B(\mathrm{c}(\mathfrak{p}), \frac{1}{4}D^{\mathrm{s}(\mathfrak{p})})}\,. \tag{42}$$

With the above definitions, it can be shown that the finitary Carleson theorem reduces to the discrete Carleson theorem.

Theorem 2.2.7 (discrete Carleson)**.** *Let* $(\mathcal{D}, c, s)$ *be a grid structure and*

$$(\mathfrak{P}, \mathcal{I}, \Omega, \mathcal{Q}, \mathrm{c}, \mathrm{s})$$

a tile structure for this grid structure. Define for every tile $\mathfrak{p} \in \mathfrak{P}$

$$E(\mathfrak{p}) = \{x \in \mathcal{I}(\mathfrak{p}) : Q(x) \in \Omega(\mathfrak{p}), \sigma_1(x) \leq \mathrm{s}(\mathfrak{p}) \leq \sigma_2(x)\} \tag{43}$$

and

$$T_{\mathfrak{p}} f(x) = \mathbf{1}_{E(\mathfrak{p})}(x) \int K_{\mathrm{s}(\mathfrak{p})}(x, y) f(y) e(Q(x)(y) - Q(x)(x))\, d\mu(y). \tag{44}$$

Then there exists a Borel set G' *with* $2\mu(G') \leq \mu(G)$ *such that for all Borel functions* $f : X \to \mathbb{C}$ *with* $|f| \leq \mathbf{1}_F$ *we have*

$$\int_{G \setminus G'} \left| \sum_{\mathfrak{p} \in \mathfrak{P}} T_{\mathfrak{p}} f(x) \right| \mathrm{d}\mu(x) \leq \frac{2^{442a^3}}{(q-1)^5} \mu(G)^{1-\frac{1}{q}} \mu(F)^{\frac{1}{q}}. \tag{45}$$

The proof of this last result is the heart of the matter, and is very technical. The full proof can be found in Sections 5, 6 and 7 of the blueprint [Bec+24c]; let us give an idea of the work required.

Firstly, cubes are classified by what fraction of them is covered by the set G. This means that we partition the set of cubes in collections $\mathcal{C}(k)$ for which this fraction is in the interval $(2^{-(k+1)}, 2^{-k}]$. This classification lifts via the function $\mathcal{I}$ to the tiles. Secondly, we make a subclassification of tiles $\mathfrak{p}$ by the fraction of the points x in $\mathcal{I}(\mathfrak{p})$ for which $d_{\mathfrak{p}}(Q(x), \mathcal{Q}(\mathfrak{p}))$ is small. As a further refinement, some tiles are *marked*, and we classify tiles by how many marked tiles are "near" a given tile. Let us note that these characterizations also take into account the orders on cubes and tiles. The order on tiles is given by $\mathfrak{p} \leq \mathfrak{p}'$ iff $\mathcal{I}(\mathfrak{p}) \leq \mathcal{I}(\mathfrak{p}')$ and $\Omega(\mathfrak{p}) \supset \Omega(\mathfrak{p}')$.

In each class in the final classification, we mark certain tiles as so-called *tree tops*. Then we eliminate 5 separate subclasses of tiles from the class. Four of these subclasses are antichains with respect to the aforementioned order on the tiles. The main result of Section 6 of the blueprint gives good bounds for the Carleson operator on an antichain $\mathfrak{A} \subseteq \mathfrak{P}$, i.e. for the operator

$$\sum_{\mathfrak{p} \in \mathfrak{A}} T_{\mathfrak{p}}.$$

The fifth subclass is small, and we can choose G' so that it contains $\mathcal{I}(\mathfrak{p})$ for all tiles $\mathfrak{p}$ in this subclass.

The remaining tiles can be partitioned into a small number of so-called *forests* (not in the graph-theoretic sense) with trees that each contain one of the aforementioned tree tops. The main result of Section 7 of the blueprint estimates the Carleson operator on such a forest. Section 5 gives the characterization of tiles, and puts these results together to prove the discrete Carleson theorem.

2.3. **Stating the main results in Lean.** Starting from the material already available in Lean's mathematical library `Mathlib`, it is easy to state the classical Carleson theorem in Lean.

```
theorem classical_carleson {f : ℝ → ℂ}
  (cont_f : Continuous f) (periodic_f : f.Periodic (2 * π)) :
  ∀ᵐ x, Filter.Tendsto (partialFourierSum · f x) Filter.atTop (nhds (f x))
```

The statement of the hypotheses ($f\colon \mathbb{R} \to \mathbb{C}$ is a continuous 2π-periodic function) closely resembles natural language, but understanding the conclusion requires some familiarity with Lean and `Mathlib`. There, $\forall^m x$ means 'for every real number x away from a set of measure zero', and `Filter.Tendsto (partialFourierSum · f x) Filter.atTop (nhds (f x))` is Lean's idiomatic way to express that the sequence $S_N f(x)$ of N-th partial Fourier sums converges to $f(x)$ when N tends to infinity (see [AM26, §11.1] for more information on `Mathlib`'s use of filters to represent limits).

The definition `partialFourierSum` encodes the N-th partial Fourier sum of $f : \mathbb{R} \to \mathbb{C}$. Here, `Finset.Icc`[4] denotes the finite set of integers $\{-N, -N+1, \cdots, N\}$, `fourierCoeffOn Real.two_pi_pos f n` is the n-th Fourier coefficient of f on $[0, 2\pi]$, and `fourier n (x : AddCircle (2 * π))` denotes e^{inx}.

```
def partialFourierSum (N : ℕ) (f : ℝ → ℂ) (x : ℝ) : ℂ :=
  Σ n ∈ Finset.Icc (-(N : ℤ)) N,
    fourierCoeffOn Real.two_pi_pos f n * fourier n (x : AddCircle (2 * π))
```

On the other hand, in order to formalize the *statements* of the metric and linearized versions of Carleson's theorem, we first needed to formalize a good amount of new concepts, starting with the definition of doubling metric measure space (see Definition 2.2), which we denote by `MeasureTheory.DoublingMeasure`. The constant A appearing in this definition will typically be 2^a.

```
class MeasureTheory.Measure.IsDoubling {X : Type*} [MeasurableSpace X]
    [PseudoMetricSpace X] (μ : Measure X) (A : outParam ℝ≥0) : Prop where
  measure_ball_two_le_same : ∀ (x : X) r, μ (ball x (2 * r)) ≤ A * μ (ball x r)
  export IsDoubling (measure_ball_two_le_same)

class MeasureTheory.DoublingMeasure (X : Type*) (A : outParam ℝ≥0)
    [PseudoMetricSpace X] extends
  CompleteSpace X, LocallyCompactSpace X, MeasureSpace X, BorelSpace X,
  IsLocallyFiniteMeasure (volume : Measure X),
  Measure.IsDoubling (volume : Measure X) A, NeZero (volume : Measure X) where
```

Next, we need to introduce the family Θ of compatible functions satisfying the properties (10), (11), (12), (13), and (14). We do this in two steps, by first creating a class `FunctionDistances` which bundles a family of continuous functions and then endowing these functions with the five properties in the class `CompatibleFunctions`. The assumption that the family is cancellative is added as a `Prop`-valued class `IsCancellative`.

Something to note here is that we created type synonyms to be able to endow Θ with different metric space structures, each coming from a ball in X, and we provided the custom notation `dist_{x, r}` for the distance corresponding to the ball $B(x, r)$.

[4] In any preorder α, one can define intervals which on each side can be either open (o), closed (c) or infinite (i). For instance, `Set.Ico a b` denotes the inverval $[a, b)$ and `Set.Iic b` denotes $(-\infty, b]$. If moreover α is a `LocallyFiniteOrder` (i.e., all bounded intervals in α are finite), `Finset.Icc`, `Finset.Ico`, `Finset.Ioc` and `Finset.Ioo` provide `Finset` versions of these intervals.

```
variable {k X : Type*} {A : ℕ} [RCLike k] [PseudoMetricSpace X]

def WithFunctionDistance (x : X) (r : ℝ) := Θ X

instance {x : X} {r : ℝ} [d : FunctionDistances k X] :
  PseudoMetricSpace (WithFunctionDistance x r) := d.metric x r

notation3 "dist_{" x ", " r "}" => @dist (WithFunctionDistance x r) _
```

With this notation, the Lean statement of the compatibility properties is quite close to the informal one. For instance, equation (12) is written as

```
cdist_mono {x₁ x₂ : X} {r₁ r₂ : ℝ} {f g : Θ}
  (h : ball x₁ r₁ ⊆ ball x₂ r₂) : dist_{x₁, r₁} f g ≤ dist_{x₂, r₂} f g
```

The class `IsOneSidedKernel` captures the notion of one-sided Calderón–Zygmund kernel (Definition 2.3).

```
class IsOneSidedKernel (a : outParam ℕ) (K : X → X → ℂ) : Prop where
  measurable_K : Measurable (uncurry K)
  norm_K_le_vol_inv (x y : X) : ‖K x y‖ ≤ C_K a / Real.vol x y
  norm_K_sub_le {x y y' : X} (h : 2 * dist y y' ≤ dist x y) : ‖K x y - K x y'‖ ≤
    (dist y y' / dist x y) ^ (a : ℝ)⁻¹ * (C_K a / Real.vol x y)
```

To avoid constant repetition of hypotheses, we create a class `KernelProofData` which contains data used in most of chapters 2 through 7 of the blueprint: a doubling measure on X, a natural number greater than three, a collection of compatible functions, and a one-sided Calderón–Zygmund kernel. See §5.2 for a more detailed discussion of this design choice.

```
class KernelProofData {X : Type*} (a : outParam ℕ) (K : outParam (X → X → ℂ))
    [PseudoMetricSpace X] where
  d : DoublingMeasure X (defaultA a)
  four_le_a : 4 ≤ a
  cf : CompatibleFunctions ℝ X (defaultA a)
  hcz : IsOneSidedKernel a K
```

The statement of the metric Carleson theorem also requires fixing two positive real numbers q, q' satisfying the equality $q^{-1} + q'^{-1} = 1$, encoded by definition `HolderConjugate` from Mathlib.

We have now described most of the prerequisites for understanding the Lean statement of Theorem 2.2.3, available as `metric_carleson`. The hypothesis `hT` relies on the definition `HasBoundedStrongType` to express the bound (25) for the `nontangentialOperator`.

```
theorem metric_carleson {X : Type*} {a : ℕ} [MetricSpace X] {q q' : ℝ≥0}
  {F G : Set X} {K : X → X → ℂ} [KernelProofData a K] {f : X → ℂ}
  [IsCancellative X (defaultτ a)] (hq : q ∈ Ioc 1 2)
  (hqq' : q.HolderConjugate q') (mF : MeasurableSet F) (mG : MeasurableSet G)
  (mf : Measurable f) (nf : (‖f ·‖) ≤ F.indicator 1)
  (hT : HasBoundedStrongType (nontangentialOperator K · ·) 2 2 volume volume
    (C_Ts a)) :
  ∫⁻ x in G, carlesonOperator K f x ≤
```

```
    C1_0_2 a q * volume G ^ (q' : ℝ)⁻¹ * volume F ^ (q : ℝ)⁻¹
```

The `carlesonOperator` is defined as a special case of the `linearizedCarlesonOperator`; these definitions correspond to those introduced in equations (21) and (20), respectively. The expression $\|\texttt{carlesonOperatorIntegrandK(Qx)R}_1\texttt{R}_2\texttt{fx}\|_{\texttt{e}}$ denotes the extended norm of the integral appearing in Equation (20), taking values in $\overline{\mathbb{R}_{\geq 0}}$ (see 5.4 for more details). In `carlesonOperatorIntegrand` below, `Set.Annulus.oo x R₁ R₂` denotes the set of real numbers y such that $R_1 < \mathrm{dist}(x, y) < R_2$.

```
def carlesonOperatorIntegrand [FunctionDistances ℝ X] (K : X → X → ℂ)
    (θ : Θ X) (R₁ R₂ : ℝ) (f : X → ℂ) (x : X) : ℂ :=
  ∫ y in Annulus.oo x R₁ R₂, K x y * f y * exp (I * θ y)

def linearizedCarlesonOperator [FunctionDistances ℝ X] (Q : X → Θ X)
    (K : X → X → ℂ) (f : X → ℂ) (x : X) : ℝ≥0∞ :=
  ⨆ (R₁ : ℝ) (R₂ : ℝ) (_ : 0 < R₁) (_ : R₁ < R₂),
    ‖carlesonOperatorIntegrand K (Q x) R₁ R₂ f x‖ₑ

def carlesonOperator [FunctionDistances ℝ X] (K : X → X → ℂ) (f : X → ℂ)
    (x : X) : ℝ≥0∞ :=
  ⨆ (θ : Θ X), linearizedCarlesonOperator (fun _ ↦ θ) K f x
```

The statement of the `linearized_metric_carleson` theorem (Theorem 2.2.4) should now be easy to parse. It requires an extra input in the form of a Borel function $Q : X \to \Theta$ with finite range, and its conclusion involves the `linearizedCarlesonOperator` T_Q. The hypothesis `hT` encodes equation (27), written in terms of the `linearizedNontangentialOperator`.

```
theorem linearized_metric_carleson {X : Type*} {a : ℕ} [MetricSpace X]
  {q q' : ℝ≥0} {F G : Set X} {K : X → X → ℂ} [KernelProofData a K]
  {Q : SimpleFunc X (Θ X)} {f : X → ℂ} [IsCancellative X (defaultτ a)]
  (hq : q ∈ Ioc 1 2) (hqq' : q.HolderConjugate q') (mF : MeasurableSet F)
  (mG : MeasurableSet G) (mf : Measurable f) (nf : (‖f ·‖) ≤ F.indicator 1)
  (hT : ∀ θ : Θ X, HasBoundedStrongType
    (linearizedNontangentialOperator Q θ K · ·) 2 2 volume volume (C_Ts a)) :
  ∫⁻ x in G, linearizedCarlesonOperator Q K f x ≤
    C1_0_2 a q * volume G ^ (q' : ℝ)⁻¹ * volume F ^ (q : ℝ)⁻¹ := by
```

2.4. **Verifying the definitions and statements.** The Lean kernel checks that all proofs are correct. It cannot ascertain that all definitions and theorems match their definition in the blueprint; this requires manual effort. One strategy (going back to at least the Liquid Tensor Experiment [Top22]) is to prove interesting theorems about these definitions (such as Carleson's classical theorem, Theorem 2.2.1). Another aspect is making the relevant definitions easy to audit: to this end, we have collected all definitions and set-up necessary for stating the main theorems, as well as the main theorems' statements, into a single file. Auditing these 170 lines of Lean code (on top of `Mathlib`) ensures the theorems say what they claim — two orders of magnitude smaller than the entire project. In addition, about a quarter of these lines are basic definitions that will be added to `Mathlib` soon — which implies getting careful review (and reduces the need to trust code beyond `Mathlib` further).

Lean ensures that the formalized proof matches the given statement. For maximum robustness (also suitable against adversarial code), we should double-check the theorem statement's meaning is not changed (via overriding notation or abuse of Lean's metaprogramming facilities). We achieve this in two ways. Firstly, each theorem is a Lean definition (with the proof constructing a term of that type) — this way, the theorem statements only require auditing the same 160 lines above. Secondly, the *comparator* tool[5] can verify such a correspondence of a given proof to a provided theorem, in a matter that is robust against malicious code or abuse of Lean's metaprogramming facilities. It is the gold standard for verification of formalisations. We have also verified the classical Carleson theorem (Theorem 2.2.1) and its metric space (Theorem 2.2.3) and linearized metric space generalizations (Theorem 2.2.4) using comparator.

3. Project organization

Planning for the Carleson project started in Fall 2023 by F.v.D. and the harmonic analysis group in Bonn, consisting of L.B., Christoph Thiele and Rajula Srivastava. At that time L.B., Asgar Jamneshan, Rajula Srivastava and Christoph Thiele had written a preprint of the proof, that turned into [Bec+24b]. This preprint consisted of a 30-page proof of Theorem 2.2.3. While the proof was precise and written with a lot of care, this paper had as intended audience experts in harmonic analysis. This meant that the proof omitted many arguments that are routine to experts, but which would require quite some work to figure out for outsiders.

After a very brief attempt to formalize the paper directly, we realized we needed a detailed blueprint that could be used as a foundation for the formalization. The goal of the blueprint was that a non-expert in harmonic analysis could read the statement and proof of any lemma in the blueprint, along with all the definitions and lemmas that were explicitly referenced by that lemma, and then would have enough information to understand the lemma and its proof. This understanding is crucial to be able to formalize it in Lean. The precise way that this lemma fits into the rest of the proof was usually not specified, and is not necessary for formalizing the proof of a single lemma.

The proof in the blueprint was divided into 179 lemmas. To prepare for the collaborative formalization effort, F.v.D. formalized the statements of most lemmas and the definitions referred to in these lemmas. This ensured a consistent statement of these lemmas, and would make it easier to find the relevant definitions for the contributors. To aid with this last point, we also manually maintained a table that recorded the correspondence between notation and definitions in the blueprint on the one side, and Lean notation and Lean declarations on the other side.

During the public formalization effort, we used three platforms for communication and coordination, an organizational structure that has also been used by other Lean projects, such as the PNT+ project[6] and the Equational Theories project [Bol+25]. Firstly, we used GitHub to have a centralized version of the code, using pull requests to ensure that at least one other contributor reviewed and approved additions or changes to the code. Secondly, we had a webpage that contained an HTML version of the blueprint, built using Patrick Massot's `leanblueprint` software.[7] This software generates a dependency graph, which

[5] https://github.com/leanprover/comparator

[6] https://alexkontorovich.github.io/PrimeNumberTheoremAnd/

[7] https://github.com/PatrickMassot/leanblueprint

gives a nice and quick overview of the progress of the project. It also generates for each lemma in the blueprint convenient links to the corresponding Lean lemma(s).

Lastly, we used a public channel of Lean's Zulip chat to discuss the project. We used this to post tasks needed for the formalization. Most of these tasks consisted of formalizing a single lemma, but some lemmas were split into multiple tasks, and some tasks consisted of something else, like refactoring existing code in either the Carleson project or Mathlib. We also used Zulip to keep track of the status of these tasks: whether they were available, claimed or completed. Zulip was also used in case someone had questions about a task, or found a potential issue in the proof of a lemma. In that case, another contributor or a member of Bonn's harmonic analysis group could help clarify the proof. In a few cases, an inaccuracy was found that required changes to the statement of one or more lemma statements in the blueprint, as described in §4.4.

We divided the material to be formalized into two categories: general material that would be useful in many other formalization projects, and specific material to the current formalization, that would likely see little reuse. We put the former material in a separate `ToMathlib` subfolder, emphasizing the aim to upstream this material to `Mathlib`. We also ensured as much as we could that these concepts and theorems would be developed in a greater generality than what was needed to formalize the current result, so that it would have an appropriate generality for `Mathlib`.

For the material specific to this formalization, we relaxed some of the contribution requirements a bit. This made the barrier of entry to contribute to the Carleson project low. Indeed, some authors of this paper had not contributed to `Mathlib` or another collaborative formalization project prior to contributing to the Carleson project.

When the project was finished, it comprised 35.6k lines of code (excluding blank lines and comments), of which 9.6k were in the `ToMathlib` subfolder (using commit `32c2adc4`).

4. Working with a blueprint

4.1. **The blueprint writing process.** Before the formalization started, the blueprint [Bec+24c] was written by the harmonic analysis group in Bonn, forming the mathematical basis of the formalization. When the formalization project was announced, the blueprint was not finished in its entirety. Some parts of Section 7 of the blueprint, which deals with arguably the most complicated case of the proof of Theorem 2.2.7, were still being finalized. Section 3, which reduces Theorem 2.2.3 to Theorem 2.2.6, had to still be written in its entirity. Most of the blueprint was finalized, and contained more than enough material to start the formalization.

The main objective of the blueprint was to provide more detailed proofs for the lemmas and propositions needed for the Carleson theorem. Another objective was to include a detailed proof of the reduction of Theorem 2.2.3 to Theorem 2.2.1. Finally, we wanted to streamline the argument to simplify the formalization. One part of the formalization that we were worried about is the so-called "invisible mathematics" that one has to formalize. These are arguments that are not written on paper, but have to be done by the formalizer to arrive at a formal proof.[8] Examples include proofs of integrability or measurability of side-goals, which are required when manipulating expressions involving integrals. To simplify these arguments, the blueprint was written to rely on finitary arguments. The main part of the proof was done in a finite setting: there are only finitely many cubes in the grid structure

[8]There is also invisible mathematics that is performed by the proof assistent, such as interpreting the meaning of notation or the scope of variables, which we will not discuss here.

and only finitely many tiles in the tile structure. Moreover, we integrate over a subset of finite measure, and the input function is a bounded whose support has finite measure. Only in the final steps of the proof, multiple limiting arguments are used to derive Theorem 2.2.3 from its finitary version, Theorem 2.2.6. This approach also had downsides, as discussed later in this section.

4.2. **Changes and refinements.** Throughout the formalization process, most of the blueprint only needed minor changes, such as fixing typos or clarifying minor details. Let us highlight four aspects where formalization resulted in more substantial changes or refinements.

4.2.1. *$I \leq J$ vs. $I \subset J$.* The axioms for a grid structure are meant to convey that the scale of a cube roughly specifies the size of a cube. Indeed, Equation (36), together with the fact that $D \gg 16$, states that if we have cubes I and J with different scales, then they are subsets/supersets of balls with drastically different radii. More precisely, if $I \subseteq J$ and $s(I) < s(J)$, then $s(I)$ is a subset of a ball with $4D^{s(I)}$ and $s(J)$ is a superset of a ball with radius $\frac{1}{4}D^{s(J)} \gg 4D^{s(I)}$. It is therefore tempting to conclude that J must be a strict superset of I. This might not be the case in an arbitrary metric space, since there is no guarantee that there are points in an annulus with specified inner and outer radii.

This also means that for two cubes I and J, if $I \subseteq J$ as sets, then we cannot assume that $s(I) \leq s(J)$. The first version of the blueprint incorrectly described the data in a cube as just a subset of X, and the scale function s as a function on these subsets. This implicitly assumed that $s(I)$ is only determined by I, viewed as a subset of X. This was fixed by including the scale as part of the data of the cube.

In the blueprint we kept using the notation "$I \subset J$" for the inequality between cubes, even though that actually included the assumption that $s(I) \leq s(J)$. This change had no further impact on downstream proofs, except for minor changes to lemma statements.

We thank Georges Gonthier for finding this inaccuracy in the blueprint.

4.2.2. *Just one top cube.* We made a second change to the definition of a grid structure, also suggested by Georges Gonthier. The first definition did not include the axiom that there exists a top cube, so it lacked the existence of I_0 and Equation (35). In the construction of the grid structure on X from Section 4 in the blueprint the top level always has a single cube, so it was free to add this to the general definition of grid structure. This slightly simplified a few proofs in later sections, and there was one computation that already implicitly assumed that there was only one top cube.

4.2.3. *The Hardy–Littlewood maximal function.* An important ingredient in the proof is a Vitali covering argument, using the Hardy–Littlewood maximal function. Recall that, given $f\colon \mathbb{R}^d \to \mathbb{C}$, the (centered) Hardy–Littlewood maximal function Mf of f is defined as

$$Mf\colon \mathbb{R}^d \to \overline{\mathbb{R}_{\geq 0}}, \quad x \mapsto \sup_{r>0} \frac{1}{|B(x,r)|} \int_{B(x,r)} |f(y)|\, dy.$$

If f is integrable, then Mf is (weak) L^1, and hence finite almost everywhere. Initially, the blueprint only used a variant taking a supremum over finitely many balls: this ensured the resulting function was *always* finite, hence real-valued. Applying it to a countable collection of balls required taking a limit and using the monotone convergence theorem. This is a somewhat inelegant solution: most results about the maximal function do hold for any

suprema, hence should be proven in the appropriate generality. We also prefer to avoid a superfluous limiting argument.

For the formal proof, we changed the definition to allow arbitrary suprema: this means the maximal function is $[0,\infty]$-valued. Being able to speak of L^p-functions valued in $[0,\infty]$ prompted the introduction of *extended norms*, see §5.4.

4.2.4. *The real interpolation theorem.* The Marcinkiewicz real interpolation theorem is a standard result in functional analysis. It is also a key prerequisite for the proof of Carleson's theorem: for example, it is applied to the Hardy–Littlewood maximal function to deduce it is of strong type L^p, for $p \in (0,\infty)$. Our formalized version aims to reduce superfluous assumptions commonly found in the mathematical literature. For instance, it applies to any sub-additive function (not merely sub-linear ones) — that is, there is no need for a scalar multiplication on the target. Similarly, any positive exponent p is allowed (as opposed to demanding $p \geq 1$).

In light of generalizing the Hardy–Littlewood maximal function, this proof had to be generalized again: applying it to the maximal functions requires a version which applies to functions with co-domain $[0,\infty]$ (or, more generally, any monoid with a continuous extended norm function). In return for a significant amount of refactoring work[9], we have formalized a more general argument that is also conceptually clearer: for instance, it demonstrates that the subtractive structure on the co-domain is not used in a meaningful way. We ended up with the following version.

Theorem 4.2.1 (real interpolation theorem)**.** *Let (X,μ) and (Y,ν) be measure spaces, let E be a topological space with the structure of an additive monoid endowed with a compatible continuous extended seminorm, let F be a topological space with a continuous extended norm function. Let $p_0,p_1,q_0,q_1,p,q \in \overline{\mathbb{R}_{\geq 0}}$ such that $p_0 \in (0,q_0], p_1 \in (0,q_1], q_0 \neq q_1$ and there exists $t \in (0,1)$ such that $\frac{1}{p} = \frac{1-t}{p_0} + \frac{t}{p_1}$ and $\frac{1}{q} = \frac{1-t}{q_0} + \frac{t}{q_1}$. Let $T : (X \to E) \to (Y \to F)$ be an operator that is subadditive in the following sense. There exists a constant $A \geq 1$ such that for $f,g \in L_0^p(X;E) \cup L_1^p(X;E)$, we have $\|T(f+g)(x)\| \leq A \cdot (\|Tf(x)\| + \|Tg(x)\|)$ for μ-a.e. $x \in E$. Furthermore, we assume that for any $f \in L^p(X;E)$, Tf is ν-a.e. strongly measurable.*

Then, if T has both weak type (p_0,q_0) and weak type (p_1,q_1), T has strong type (p,q).

For further detail, we refer the reader to a separate article [PR].

4.3. **Constant tweaking.** As explained below in §5.1, the constants appearing in the blueprint are always explicit and most often in the form 2^{na^3} where a is a given parameter of the system and n is a concrete natural number, ranging from 1 or 2 in the first lemmas to 443 in the final result.

When a result depends on several previous lemmas, its constant can typically be expressed in terms of the previous lemmas, giving chains of conditions for the overall validity of the constants. When a minor mistake is noticed in the proof of a lemma, increasing slightly its constant, this mandates corrections in all the statements that depend transitively on it. The standard practice in paper mathematics to avoid this kind of intricate dependency is not to care about the constants, and formulate the lemmas as "there exists a universal constant C such that ...". This is not a viable approach in formalized mathematics, as explained in §5.1.

[9] the initial formalization was about 5000 lines long; the generalization touched about 2000 of them

To mitigate this issue and avoid endless corrections, during the development process the constant in each lemma did not have a numerical value, but instead it was expressed as a function of the constants in the other lemmas. Therefore, changing the constant in Lemma A would automatically change the constant of Lemma B that depends on Lemma A, not altering the validity of its proof. At the very end of the formalization process, when all the proofs worked, we could finally register the numerical values of the constants with a correctness guarantee, starting with the first lemmas and propagating the values iteratively. The blueprint was then fixed using these computer-checked constants – and indeed many constants written in earlier versions of the blueprint were not correct (some of them too optimistic, some of them too pessimistic), although this had absolutely no consequence on the validity of the proof strategy.

4.4. **Dealing with inaccuracies.** Our formalization of the metric Carleson theorem certified that the main result in [Bec+24b] was correct, and that its detailed proof provided in [Bec+24c] did not contain any serious mathematical mistakes. However, it did contain several errors and imprecisions, most of which were not present in the original paper; instead they were introduced during the blueprint writing process, which required to provide a much higher level of detail than in a standard harmonic analysis paper. Moreover, many proofs in the blueprint relied on non-standard finitary arguments, which sometimes led to proofs that were not correct in the edge cases.

In most cases, the mistakes detected in the blueprint were minor and could be resolved by the formalizers themselves, for instance by adding a missing hypothesis or by modifying a certain constant. Most of these errors were local in nature, meaning that they only affected the proof of a certain lemma, or at most a couple of results, if the lemma statement needed to be modified.

However, fixing certain proofs required the assistance of harmonic analysis experts with a deep understanding of the overall proof strategy. For instance, the proof of an earlier version of [Bec+24c, Lemma 6.2.3] relied on the fact that a certain pair of dyadic cubes were not disjoint, which was not necessarily the case under the lemma assumptions. The non-obvious solution required to add an extra hypothesis that a certain pair of open balls were not disjoint, and to increase a constant appearing in the lemma statement, after which the previous proof could be adapted to conclude the result. Luckily this error did not propagate, since the only application of the lemma was still possible under the new hypotheses.

A more involved fix was required for [Bec+24c, Lemma 6.3.4], whose original proof presented several issues. The first needed change was simple, and consisted on further restricting the allowed values for the parameter ϑ, so that a certain covering property could be applied. However, the proof was also incorrect in the case where the value $s(\mathfrak{p})$ associated to a certain tile $\mathfrak{p}$ took the minimum possible value, and it incorrectly assumed that a certain new tile constructed during the proof would always be distinct from $\mathfrak{p}$. To repair these issues, the statements of both Lemma 6.3.3 and Lemma 6.3.4 had to be modified, and significant changes had to be applied to the proof of Lemma 6.3.4.

As can be expected, the formalization surfaced various small inaccuracies in the blueprint. This is not surprising, since experience shows that a human-written mathematical document of this scale usually contains at least a few minor technical flaws. At some places, proofs were slightly inaccurate, and at other places lemmas required stronger assumptions than originally stated. However, when assumptions had to be strengthened, the stronger assumptions were

always satisfied at the places where the lemmas were used. Therefore, all these issues were purely local and did not impact the global proof.

To highlight the kinds of issues that can sometimes slip through the cracks in informal mathematical arguments we give a few illustrative examples:

- Lemma 11.1.6 in the blueprint is a version of the boundedness of the Hilbert transform on $L^2(\mathbb{R})$. It is proved by localization and periodization, ultimately using that $\|S_N f\|_{L^2[0,2\pi]} \leq \|f\|_{L^2[0,2\pi]}$ for periodic functions. One step in the proof involves a convolution $\int f(y)g(x-y)\,\mathrm{d}y$ for some kernel g. In the initial argument, the position of $x-y$ was not carefully controlled, claiming to use a universal bound on g while this bound was only valid in some limited interval. The issue was fixed by truncating f and only looking at x in some interval to make sure that, when $f(y) \neq 0$, then $x-y$ belonged to the place where g is well controlled. These additional assumptions were satisfied in the rest of the proof.
- Lemma 8.0.1 in the blueprint is a regularization lemma, saying that some Hölder functions can be approximated by Lipschitz functions, constructed through a smoothing integral formula. The original proof assumed implicitly that a continuous function which is τ–Hölder–continuous in a closed ball $B(z,R)$ and supported there is globally τ–Hölder–continuous with the same control. This is true in well-behaved spaces, for instance in $\mathbb{R}^n$, but not in the generality of the paper. For instance, if the annulus $\{y \mid d(x,y) \in [R-\epsilon, R+\epsilon]\}$ is empty, then the characteristic function of $B(z,R)$ is Hölder–continuous on the ball with a uniform bound and supported there, but its extension to the whole space has a norm which may grow like $\epsilon^{-\tau}$. The fix was to modify the assumptions of the lemma, assuming from the start that the function is τ–Hölder–continuous in the larger ball $B(z,2R)$. This more restrictive assumption is satisfied in all the places where the lemma is used, although this has forced us to alter some constants in lemma statements.

4.5. **Lessons learned.** In hindsight, in some places slightly different design choices during the writing of the blueprint could have further streamlined the formalization process.

One of these was the use of finitary arguments, specifically the non-standard definition of the Hardy–Littlewood maximal function over finite collections of balls, and the decision to only allow finitely many scales in the constructions of grid and tile structures.

While these choices were made partly with the intent to simplify, they ultimately led to additional technical complications in the blueprint, causing several inaccuracies and some friction during formalization.

In the case of the maximal function, the finitary formulation required explicitly specifying various finite collections of balls, some of which were initially slightly off. As mentioned above, the finitary variant also led to an unnecessary limiting argument to generalize to the standard version. The finite grid constructions on the other hand meant that several edge cases had to be handled when encountering the smallest or largest scale. This introduced additional formalization effort and led to small inaccuracies in the blueprint again causing some confusion during formalization.

- Refer to general results early on.
- Generalize during blueprint writing.

An additional minor point was that the lack of definition environments in the blueprint also meant that definitions were absent from the dependency graph.

5. Design decisions

5.1. **Treatment of constants.** In analysis papers, one often needs to compare the growth of two functions $f, g : X \to Y$, where X, Y are real or complex normed vector spaces. This is often expressed by statements of the form

$$\exists C > 0, \forall x \in X, \|f(x)\| \leq C\|g(x)\|. \tag{46}$$

In many cases, one only cares about the existence of the positive constant C, but not about its concrete value. The constant is often left implicit, by writing (46) as

$$\forall x \in X, \|f(x)\| \lesssim \|g(x)\|.$$

This was also the case in the first version of [Bec+24b]. However, while Lean allows one to work with existential statements such as (46), this would have made the formalization harder. From the mathematical point of view, it would require the formalizers to figure out how large these constants needed to be and how the constants required to make each lemma hold were related to each other. From the technical side, it would require to work with witnesses for these existential statements, which can introduce some overhead. Therefore, while they were writing the blueprint, the authors of [Bec+24b] decided to make all of the constants explicit. At the beginning of the blueprint, they fixed a real number $1 < q \leq 2$ and a natural number $a \geq 4$, which is used to define the constants $D = 2^{100a^2}, \kappa := 2^{-10a}$ and $Z := 2^{12a}$. The bounding constants that appear in the blueprint theorem statements are expressed as functions of these fundamental constants.

In the formalization, we introduced a constant `c` with a value of 100, coming from the exponent of D. Every other constant is defined in terms of a, q and `c`, and we use the notation convention `Cx_y_z` to denote the constant appearing in result `x.y.z` of the blueprint.

As a side effect of the formalization project, once the main result had been formalized we were able to check that, replacing $D := 2^{100a^2}$ by 2^{7a^2}, the results still hold without needing to make any adjustments to the proofs. This improves the constant appearing in the main theorem by an order of magnitude.

5.2. **Standing assumptions and the `ProofData` pattern.** Mathematical papers often contain standing assumptions, that are in place for all the statements in the paper, or in a specific section. This avoids tedious repetition in the statements, sometimes at a slight expense in readability as the reader has to locate the places where such standing assumptions are made. Such a practice is especially important when the standing assumptions are lengthy and technical.

The standing assumptions for the statement of the generalization of Carleson's theorem are quite formidable. We will not repeat them here, see [Bec+24c, Theorem 1.1.1] for details. Let us just mention that they involve a metric space X, a measure μ on X which is doubling for a constant 2^a, a family Θ of functions from X to $\mathbb{R}$, a distance on Θ satisfying five properties as well as a cancellativity condition, and a kernel $K \colon X \times X \to \mathbb{R}$ with Calderón–Zygmund like properties. The proofs involve even more additional data, notably two bounded measurable sets F and G, an integrability exponent $q \in (1, 2]$ and two measurable functions σ_1 and σ_2 with finite range.

One possibility to solve this issue in the formalization would be to say that there is no issue, and include as assumptions in each lemma the minimal subset of the standing assumptions that is necessary for the statement to make sense. This would come at a high readability

cost. Moreover, each time one applies such a lemma one would need to provide the individual assumptions as a long list of parameters to the lemma.

It is a better idea to try to import in the formalization the practice of standing assumptions, in some form. One could define a structure containing all these standing assumptions, and have it as a parameter in every lemma and every definition. The readability cost would be minor, but still there. In the Carleson project, we have used another strategy, making all the standing assumptions completely implicit — in particular, we do not need to pass them between lemmas. The implementation takes advantage of a feature of the language, typeclasses, initially designed for a completely different purpose.

The way to declare that a space is a metric space is the following line:

```
variable {X : Type*} [MetricSpace X]
```

The declaration between square brackets is the typeclass. When one writes the distance `dist x y` between two points x and y of X, the system tries to locate such a metric space typeclass instance on X, and uses the distance provided by this instance. The same system is used to provide algebraic operations on types, and properties of these.

Typeclasses can be used to capture standing assumptions as follows. Let us declare a new typeclass, of the form `[KernelProofData X]`, containing not a distance as in the above example, but a measure on X, a doubling constant, a family Θ of functions from $X \times X$ to $\mathbb{R}$, and so on, i.e., all our standing assumptions. Then, whenever one tries to use an object from the standing assumptions, the systems looks for a `KernelProofData` instance and fetches the object from the typeclass instance, just like it fetches the distance in a metric space.

Another advantage of this approach is that typeclasses can extend each other (just like a normed space contains more information than a metric space). This means we can craft one typeclass containing the standing assumptions for the statements, and another extended typeclass `ProofData` with more objects for the proofs. When a `ProofData` instance is in scope but Lean needs a `KernelProofData` instance, typeclass inference automatically fetches the latter from the former, just as it transparently interprets a normed space as a metric space.

The specific implementation of this idea in the Carleson project is slightly more involved, as it is written in the form `[ProofData a q K sigma1 sigma2 F G]`. However, all the additional data a, q, K, σ_1, σ_2, F and G are associated automatically and uniquely to X (in technical terms, they are *outparams*). This means that the user never has to give one of them explicitly, as we expect of a suitable system to handle standing assumptions.

5.3. **Working with real numbers.** The project differentiated between three different types of (extended) real numbers, $\mathbb{R}$, $\mathbb{R}_{\geq 0}$ and $\overline{\mathbb{R}_{\geq 0}}$, which gave necessary benefits but also created difficulties in the formalization and tension in design decisions. Indeed, in which of the three types to put constants, subsets, integrands, and perform computations?

A term x of type $\mathbb{R}_{\geq 0}$ is not also of type $\mathbb{R}$, nor is it of type $\overline{\mathbb{R}_{\geq 0}}$, even though in set-based mathematics there are inclusions of the corresponding sets. In type-theory however, one needs coercions, or casts, to go from one type to another. These coercions are often inserted and applied automatically. The `norm_cast` tactic can be really useful to normalize such casts.

However, one also needs conversions in the other directions: sometimes one would need to convert real numbers to nonnegative real numbers. The conversions often play together with other operations, in that one for instance would need to know if, and often under which conditions, the product of the conversions is the conversion of the product. Indeed, since

`Real.toNNReal` maps negative numbers to zero, we get that $\texttt{Real.toNNReal}(-2) * \texttt{Real.toNNReal}(-3) \neq \texttt{R}$ Even with the necessary side conditions, the `norm_cast` tactic does not get very far in these situations. This can make conversions between number types quite painful, and puts some pressure to limit conversions and put as many numbers as possible in the same number type.

Sometimes one is forced to add a conversion, because certain operations are not defined on all types. Consider for instance the example of performing a computation on exponents in L^p-function spaces. Around the real interpolation theorem, one needs to compute p in terms of other exponents that naturally live in $\overline{\mathbb{R}_{\geq 0}}$. Yet there is no definition for a^p if p is in $\overline{\mathbb{R}_{\geq 0}}$, and therefore conversions from $\overline{\mathbb{R}_{\geq 0}}$ to $\mathbb{R}$ are unavoidable.

There are also advantages of differentiating between the spaces. An advantage of putting for instance a constant in `NNReal` over `ENNReal` or `Real`, is that it encodes that the constant cannot be infinity and that it needs to be non-negative.

Another advantage of differentiating comes from the different levels of tactic support. For real numbers, computations are much easier to perform, using the `linarith` and `field_simp` tactics. For terms in $\mathbb{R}_{\geq 0}$, support in `linarith` was only added after this project was completed.[10] Neither `linarith` nor `field_simp` support terms in $\overline{\mathbb{R}_{\geq 0}}$. The tactic `ring` *does* work for these types, but not as well, and the performance for terms in $\overline{\mathbb{R}_{\geq 0}}$ is even worse than for terms in $\mathbb{R}_{\geq 0}$. For both $\overline{\mathbb{R}_{\geq 0}}$ and $\mathbb{R}_{\geq 0}$, it will generally fail when a goal needs properties of subtraction: it even fails to prove that $x - x = 0$. If $x : \mathbb{R}_{\geq 0}$, the `ring` tactic can prove that $(x^{-1})^2 = (x^2)^{-1}$, or that $3/x = 3x^{-1}$, but it fails if $x : \overline{\mathbb{R}_{\geq 0}}$. Similarly `norm_num` also works for numbers in $\overline{\mathbb{R}_{\geq 0}}$, but again not as well as for real numbers, as even with the necessary side conditions it would not simplify a term of the form a/a. Both `ring` and `norm_num` fail to solve $(3/5) * (5/3) = 1$ when these numbers are viewed as numbers in $\overline{\mathbb{R}_{\geq 0}}$.

On the nose, many results about real numbers transfer to $\overline{\mathbb{R}_{\geq 0}}$ provided all arguments are finite; in many cases, these finiteness hypotheses are actually necessary. Manipulating expressions in $\overline{\mathbb{R}_{\geq 0}}$ thus often requires proving such finiteness side goals. The `finiteness` tactic (initially developed for the formalization of the PFR project) solves many such proof obligations automatically. We made this tactic more powerful, completing (for instance) support for all relevant cases in the Carleson project.

A desirable future step is adding improved interaction with the `positivity` tactic (which solves goals of the form $0 \leq x$ or $0 < x$). Proving a goal $(x + y + 1)^{-1} < \infty$ (for $x, y \in \overline{\mathbb{R}_{\geq 0}}$) requires proving that $x + y + 1$ is non-zero; this is already handled by calling `positivity`. Conversely, `positivity` alone cannot prove goals $x^{-1} \neq 0$ for $x \in \overline{\mathbb{R}_{\geq 0}}$ — as this is true only if x is finite. Allowing `positivity` to call `finiteness` would improve automation significantly.

An independently useful, more ambitious yet more powerful tactic would allow reducing goals in $\overline{\mathbb{R}_{\geq 0}}$ to $\mathbb{R}_{\geq 0}$: this could be implemented by case-splitting on each relevant variable being finite or infinite. The finite case allows reduction to $\mathbb{R}_{\geq 0}$ arithmetic; the infinite case will simplify calculations (as, for example, ∞ added to any number in $\overline{\mathbb{R}_{\geq 0}}$ is still ∞). Such goals can hopefully be proven automatically, or reduced to much simpler tasks for the user.[11] The precise design of such a tactic has been discussed (including by some of this paper's authors); its implementation is being worked on.

[10] https://github.com/leanprover-community/mathlib4/pull/35155

[11] For multiplication, as $0 * \infty = 0$, simplifying $a * \infty$ to ∞ involves splitting cases by whether a is zero or non-zero. Lest the reader worry about combinatorial explosion, let us remark that (1) splitting on a should mean splitting on the cases of a being infinite is not necessary any more, and (2) such splits can be performed as required.

As a final example around considerations on which number types to use, consider the supremum of a set. As one eventually is anyway interested in situations in which this supremum is finite and the set is bounded from above, one could consider a setup in which the sets considered are subsets of real numbers and the suprema are real numbers as well. This was the choice at the outset of the project. However, in this setup, one needs to constantly carry around a proof or assumption that the set is indeed bounded. This is cumbersome, and therefore it turned out to be much better to let suprema just take values in the `ENNReal`s.

Gradually, we came to the conclusion that, with some exceptions, it was best to use $\overline{\mathbb{R}_{\geq 0}}$ as much as possible.

5.4. **Use of `ENorm`.** This project motivated the creation of a new formalization abstraction: *extended norms* (`enorm`s for short) generalize many results from normed spaces to $\overline{\mathbb{R}_{\geq 0}}$. Their introduction was motivated by defining the Hardy–Littlewood maximal with codomain $\overline{\mathbb{R}_{\geq 0}} = [0, \infty]$ (see §4.2). Previously, a statement "$f \in L^p$" was only defined for functions $f \colon X \to E$ from a measure space X to a normed space E. Speaking about the maximal function being in weak L^p requires a notion of L^p functions with target $\overline{\mathbb{R}_{\geq 0}}$. Fortunately, many basic facts about L^p functions do not require the target to be a normed space and have reasonable analogues for $\overline{\mathbb{R}_{\geq 0}}$: for instance, the L^p norm of $f \colon X \to \overline{\mathbb{R}_{\geq 0}}$ for $p < \infty$ can be defined as $(\int_X |f(x)|^p \, \mathrm{d}x)^{1/p}$, where the integrand is the function $X \to [0, \infty], x \mapsto |f(x)|^p$.

Motivated by this observation, we introduced a new notation class `ENorm`, describing a type endowed with a function `enorm`$\colon X \to \overline{\mathbb{R}_{\geq 0}}$. This captures both normed spaces and $\overline{\mathbb{R}_{\geq 0}}$. The enorm of a normed space is the ambient norm, considered as a function into $\overline{\mathbb{R}_{\geq 0}}$; the enorm on $\overline{\mathbb{R}_{\geq 0}}$ is the identity function. Some definitions only require the existence of an enorm; many results require suitable compatibility conditions. Most lemmas necessary in this project only require an `ENormedAddCommMonoid`, i.e. an abelian monoid endowed with a continuous enorm that is positive definite and satisfies the triangle inequality. Many lemmas, in fact, only require slightly weaker assumptions.

Pursuing this approach required generalizing all necessary definitions to `enorm`s, as well as all basic results that are required in this project. This was a significant effort, but the verified nature of formalization helped a lot by allowing to do so incrementally: in a first step, the new definition was added (together with the enorm instance on normed spaces). Then, all basic definitions were changed to allow an `enorm` as codomain, whenever this did not require further changes. A third step modified many lemma statements to use `enorm`s instead of norms. The final phase involved generalizing lemmas: often, this required small changes (such as, constants changing from a real to an extended real number, or adding a hypothesis about some quantity in $\overline{\mathbb{R}_{\geq 0}}$ being finite). At each point, verification ensured correctness: if all proofs still compiled, no more changes needed to be made.

To illustrate the magnitude of this refactoring, let us give some statistics: phase two (changing the basic definitions) changed about 300 lines of code; phase three (changing lemma statements to enorms) about 1000, and the lemma generalizations so far touched about 2500 lines of code.

The last phase (generalizing mathlib lemmas) is not exhaustive: while substantial parts of mathlib have been generalized, some areas would require further refactor (and will follow as needed). To give an example, the total variation of a path in $\overline{\mathbb{R}_{\geq 0}}$ is a sensible quantity, even though $\overline{\mathbb{R}_{\geq 0}}$ is not a metric space. Introducing a weaker notion of extended metric spaces

will enable new applications, and also provide the proper abstraction to state many scalar multiplication lemmas nicely. This is an ongoing effort by Felix Pernegger.

The process of generalization revealed that this abstraction is also useful on its own: previously, many proofs in mathlib involved converting between norms taking values in $\mathbb{R}$ or $\mathbb{R}_{\geq 0}$; using enorms provides a useful alternative. At the same time, it illustrates how the right abstractions can avoid duplicating work, while providing a useful clarification to mathematics.

5.5. **Working with L^p functions.** Working with L^p functions presents a design choice: should one work with members of L^p space (i.e., equivalence classes of functions up to almost everywhere equality, `MeasureTheory.Lp`), or work with representatives (`MemLp`) instead? The mathematical difference is very small: virtually all operations in the project respect a.e. equality. Formalization ergonomics, however, are greatly different: working with equivalence classes is much more cumbersome, as identities on the level of representatives only hold almost everywhere. (For example, $(f+g)x = fx+gx$ holds on the nose for actual functions; for representatives of $\tilde{f}, \tilde{g}$ and $\tilde{f}+\tilde{g} \in L^p$, a priori this only holds for almost every x.) While the `filter_upwards` tactic simplifies reducing results from almost every to every x, these details add up: it is easiest to omit the passage entirely by working with functions in L^p (i.e., with `MemLp` directly).

Incidentally, lemmas in Mathlib are mostly stated for `MemLp` instead of `Lp`: this underscores this choice.

This relates well with the use of enorms (5.4): `LpEp`μ as a normed space requires functions with codomain in a Banach space; this excludes e.g. $\overline{\mathbb{R}_{\geq 0}}$ which does not possess a well-behaved subtraction operation. Phrasing lemmas using `MemLp` first enables the generalization to $\overline{\mathbb{R}_{\geq 0}}$ in the first place.

This is somewhat similar to the trade-off between bundled and unbundled objects: established common wisdom is that unbundled or partially bundled objects are often better to work with than bundled ones. For instance, a continuous function $f\colon X \to Y$ would usually be formalized as `: X → Yf : X → Y (hf : Continuous f)` instead of the bundled `f : ContinuousMap X Y`.

5.6. **Test function classes.** A common task during the formalization was to prove the various integrability and measurability side goals arising when estimating integrals. As is typical in harmonic analysis, most technical estimates are *a priori* estimates involving input functions in suitable test function classes.

In many parts of this project, bounded measurable compactly supported functions provided a convenient test function class. We also used the slightly larger class of bounded measurable functions supported on a set of finite measure in places where this weaker hypothesis matches the blueprint more closely. Since most constructions appearing in the proof leave these function classes invariant, integrability and measurability considerations are usually mathematically trivial.

It is desirable for this simplicity to be reflected in the formalization. This motivated packaging the recurring side conditions into test function classes.

The structures `BoundedCompactSupport` and `BoundedFiniteSupport` contain essentially bounded measurable functions with compact support and with support of finite measure, respectively. A function f is essentially bounded if there exists a real number C such that $\|f(x)\| \leq C$ almost everywhere.

Once these hypotheses are bundled, later proofs can invoke short API lemmas instead of repeatedly reproving the same measure-theoretic facts. For example, from a hypothesis `hf : BoundedCompactSupport f` one immediately obtains that `f` is a member of any desired L^p space, or that `s.indicator f` again has bounded compact support when `s` is measurable. Lemmas such as `hf.carlesonSum` show that the relevant operators preserve the test function class, so once compact support has been established at the input, it remains available throughout the argument.

Choosing the right definition of these structures is subtle. Since most functions we're working with are only well-behaved up to almost everywhere equality (see 5.5), there is a choice in deciding that the functions are only essentially bounded or bounded everywhere. Similarly, for `BoundedCompactSupport` there is a choice in requiring that the function has compact support or essentially compact support (that is, its support is the union of a relatively compact set and a null set, where a relatively compact set is by definition a subset of a compact set).

In Carleson, we originally started with functions that are bounded everywhere and have compact support, but we switched[12] to functions that are only essentially bounded, but still have compact support. The reason for this is that Mathlib uses essentially bounded functions frequently in measure theory, phrased as `MemLp f ∞ μ`.

These notions are mathematically very similar, but the choice has impact in precisely which lemmas you can prove about it. For example, if g is bounded with compact support, and f is a closed embedding, is $g \circ f$ a bounded function with compact support? This is only true if g has compact support and is bounded everywhere; otherwise f may send a set of positive measure to a set of measure 0 where g is ill-behaved.

Another consideration into this design decision is how nicely one can work with these functions. Some arguments are a little easier if we know that a function is well-behaved everywhere, and not just almost everywhere. In the Carleson project, there were two tricky arguments using an operation `adjointCarlesonSum`. We prove that this operation is bounded with compact support, for these arguments it was convenient to use that this operation is bounded everywhere, and not just essentially bounded. While these arguments could be adapted using only that this operation is only essentially bounded, the argument consists of some tricky manipulations with suprema over the arguments, and these would have to be changed to essential suprema, which are a bit trickier to work with. In the end, we decided to prove that this function was both bounded with compact support, and additionally bounded everywhere.

These classes are also tagged with the `@[fun_prop]` attribute. `fun_prop` is a `Mathlib`-tactic developed by Tomáš Skřivan that automatically uses tagged lemmas to show that a function has a certain property (continuity, differentiability, measurability, etc.). This tactic is good at writing complicated functions as a composition of simpler functions, and then proving function properties by proving them first about the simpler functions.

`fun_prop` is not as well-suited for proving integrability as the other function properties mentioned above, since the composition and product of integrable functions need not be integrable. The class `BoundedCompactSupport` is a step towards making `fun_prop` more usable for integrability, since it is better behaved w.r.t. multiplication and composition.

[12] see pull requests 311 and 342

5.7. **Common pitfalls.** As described in 5.3, at the beginning of the project we tried to avoid conversions by using the type `Real` as much as possible, even if quantities were naturally nonnegative or possibly infinite. In the course of the project, in different contexts we realized that using `NNReal` or `ENNReal` instead would lead to shorter proofs and reduce the number of side conditions. While generally enabling simplifications, such changes not only came at the cost of additional conversions at the boundary between different contexts but also sometimes necessitated extending the API where `Mathlib` currently only supports `Real`. For instance, to be able to integrate functions on `NNReal` or `ENNReal`, we added `volume` instances on these types and proved basic properties and relations. For example, when integrating a function `f` : $\mathbb{R}_{\geq 0\infty} \to \mathbb{R}_{\geq 0\infty}$, the following expressions should be equivalent.

```
∫⁻ (x : ℝ) in Ioi 0, f (ENNReal.ofReal x)
-- TODO: write out the coercion instead!
∫⁻ (x : ℝ≥0), f (↑x)
∫⁻ (x : ℝ≥0∞), f x
```

Another case of equivalent representations we encountered was that of integrating on a set, i.e. with respect to a restricted measure, versus integrating an indicator function, again both useful for different purposes. Conversion here was straightforward using the lemmas `integral_indicator` resp. `lintegral_indicator` in `Mathlib`. Similarly, we can represent a periodic function on the real line as `f : ℝ → ℂ` with the property `f.Periodic T` and integrate it with respect to the measure `volume.restrict (Ioc 0 T) : Measure ℝ`. Alternatively, we can write it as `f : AddCircle T → ℂ` and integrate it with respect to `volume : Measure (AddCircle T)`. Again, in different context the existing `Mathlib` API was more suitable for working with one of the two versions. It is still work in progress to extend `Mathlib` with API for the other version where no version is naturally favorable, to reduce the overall number of conversions.

Finally, there were several results in the project that required us to consider sets of elements from a given finite type `T`. Since all such sets are finite, from a mathematical point of view there is no difference between the types `Set T` and `Finset T`. However, these are different types in Lean: `Set T` is the type of all sets consisting of elements of type `T`, while `Finset T` is the type of finite sets of elements of `T`. Therefore, we needed to make a decision about which of these types to use in the Carleson project. For the most part, we chose to use `Set T`, although sometimes `Finset T` is used inside proofs, or when required to apply existing `Mathlib` results. In hindsight, it would likely have been slightly more convenient to use `Finset T` throughout, to avoid issues like the following.

If `T` is endowed with a `Fintype` instance, then `Finset.sum` can still be applied to a given `s : Set T`. There are two possible syntaxes to do this, as the following example shows. Note that the displayed equality is true *propositionally* but not *definitionally*: the left hand side is a sum over `s.toFinset`, whereas the right hand side is a sum over `{p : T | p ∈ s}`, and these `Finset`s are not definitionally equal.

```
example {T : Type} [Fintype T] {s : Set T} [DecidablePred fun p ↦ p ∈ s]
  {f : T → ℕ} : ∑ p ∈ s, f p = ∑ p with p ∈ s, f p
```

6. Conclusion

The Carleson project, with a total of 28 contributors, was one of the largest collaborative formalization efforts up to date (excluding extensive mathematical libraries such as `Mathlib` or MathComp), as well as the first formalization in Lean of a state-of-the-art theorem from the area of harmonic analysis.

Formalizing the metric Carleson theorem required us to formalize foundational results from harmonic analysis, which we are submitting to `Mathlib` and which will be useful for the formalization of other results from this field. Future projects will also benefit from some of the insights we learned during the project, including the creation of extended norms and our reflections about the most convenient ways to work with real number types.

Beyond topic-specific insights, we also learned lessons from the collaborative nature of this project which apply to formalization efforts from any area of mathematics. Our way of assigning and keeping track of ongoing tasks in the Zulip chat was sub-optimal, and if we were to repeat the project, we would instead use the existing GitHub issue and project infrastructure (as is done for instance in the PNT+ project). Zulip would still be used as a discussion platform, and in particular to discuss potential inaccuracies within the blueprint proofs. To make this more efficient, we would ask one of the topic experts (in our case, an expert in harmonic analysis) to frequently monitor this chat and respond to questions, instead of having F.v.D. as an intermediary between the remote formalizers and the harmonic analysis experts.

We would also modify some aspects of the blueprint for the formalization. We would use LaTeX environments for the main definitions in the project, to facilitate their discovery, and we would try to ensure that the results in the blueprint were already proven at the right degree of generality; this would specially apply to results meant for `Mathlib`, for which we want a general statement and proof, rather than an specialization to the project setting. Finally, we would weight more carefully the benefits of using non-standard proof methods in a hope to simplify the formalization, since this ended up causing most of the mathematical inaccuracies in the first version of the blueprint.

After the Carleson project was finished, we have continued to prove further results in the Carleson repository. Theorem 2.2.3 is strong enough to prove stronger versions of the Theorem 2.2.1. As Hunt [Hun68] showed in 1968, the assumption in Theorem 2.2.1 that f is continuous can be weakened to the statement that f is L^p for $p > 1$. Proving this requires some additional techniques from analysis.

The *Lorentz-norm* of a function is

$$\|f\|_{L^{p,q}(X,\mu)} := p^{\frac{1}{q}} \|t\mu\{|f| > t\}^{\frac{1}{p}}\|_{L^q(\mathbb{R}_{>0}, \frac{dt}{t})}.$$

It is not hard to see that the $L^{p,p}$-norm corresponds to the L^p-norm and the $L^{p,\infty}$-norm corresponds to the weak L^p norm (which we also denoted $L^{p,\infty}$ before). Restricted weak-type estimates such as the one in Theorem 2.2.3 specify the boundedness of an operator with respect to the $L^{p,1}$-norm.

There is a generalization of the Marcinkiewicz interpolation theorem to Lorentz spaces, which we have not formalized yet. However, we have finished the formalization that this generalized interpolation result implies Theorem 2.2.1 for L^p functions with $p > 1$.

References

[AM26] Jeremy Avigad and Patrick Massot. *Mathematics in Lean*. 2026. URL: https://leanprover-community.github.io/mathematics_in_lean/.

[Bec+24a] Lars Becker, Floris van Doorn, Asgar Jamneshan, Rajula Srivastava, and Christoph Thiele. *Carleson Operators on Doubling Metric Measure Spaces*. 2024. eprint: 2405.06423v2.

[Bec+24b] Lars Becker, Floris van Doorn, Asgar Jamneshan, Rajula Srivastava, and Christoph Thiele. "Carleson operators on doubling metric measure spaces". arXiv:2508.05563 [math.CA]. 2024.

[Bec+24c] Lars Becker et al. "A blueprint for the formalization of Carleson's theorem on convergence of Fourier series". https://arxiv.org/abs/2405.06423. 2024.

[Bil67] P. Billard. "Sur la convergence presque partout des séries de Fourier-Walsh des fonctions de l'espace $L^2(0, 1)$". In: *Studia Math.* 28 (1966/67), pp. 363–388. ISSN: 0039-3223,1730-6337. DOI: 10.4064/sm-28-3-363-388. URL: https://doi.org/10.4064/sm-28-3-363-388.

[Bol+25] Matthew Bolan et al. *The Equational Theories Project: Advancing Collaborative Mathematical Research at Scale*. 2025. arXiv: 2512.07087 [math.RA]. URL: https://arxiv.org/abs/2512.07087.

[Car66] Lennart Carleson. "On convergence and growth of partial sums of Fourier series". In: *Acta Math.* 116 (1966), pp. 135–157. ISSN: 0001-5962. DOI: 10.1007/BF02392815. URL: https://doi.org/10.1007/BF02392815.

[Dem15] Ciprian Demeter. "A guide to Carleson's theorem". In: *Rocky Mountain J. Math.* 45.1 (2015), pp. 169–212. ISSN: 0035-7596,1945-3795. DOI: 10.1216/RMJ-2015-45-1-169. URL: https://doi.org/10.1216/RMJ-2015-45-1-169.

[Doo21] Floris van Doorn. *Formalized Haar Measure*. 2021. arXiv: 2102.07636 [cs.LO]. URL: https://arxiv.org/abs/2102.07636.

[Fef73] Charles Fefferman. "Pointwise convergence of Fourier series". In: *Ann. of Math. (2)* 98 (1973), pp. 551–571. ISSN: 0003-486X. DOI: 10.2307/1970917. URL: https://doi.org/10.2307/1970917.

[Gou22] Sébastien Gouëzel. "A Formalization of the Change of Variables Formula for Integrals in mathlib". In: *Intelligent Computer Mathematics*. Ed. by Kevin Buzzard and Temur Kutsia. Cham: Springer International Publishing, 2022, pp. 3–18. ISBN: 978-3-031-16681-5.

[Hun68] Richard A. Hunt. "On the convergence of Fourier series". In: *Orthogonal Expansions and their Continuous Analogues (Proc. Conf., Edwardsville, Ill., 1967)*. Southern Illinois Univ. Press, Carbondale, IL, 1968, pp. 235–255.

[Kol23] A. Kolmogorov. "Une série de Fourier–Lebesgue divergente presque partout". In: *Fund. Math.* 4 (1923), pp. 324–328.

[Lac04] Michael T. Lacey. "Carleson's theorem: proof, complements, variations". In: *Publ. Mat.* 48.2 (2004), pp. 251–307. ISSN: 0214-1493,2014-4350. DOI: 10.5565/PUBLMAT_48204_01. URL: https://doi.org/10.5565/PUBLMAT_48204_01.

[LT00] Michael Lacey and Christoph Thiele. "A proof of boundedness of the Carleson operator". In: *Math. Res. Lett.* 7.4 (2000), pp. 361–370. ISSN: 1073-2780. DOI: 10.4310/MRL.2000.v7.n4.a1. URL: https://doi.org/10.4310/MRL.2000.v7.n4.a1.

[Luz13] Nikolai Luzin. "Sur la convergence des séries trigonométrique de Fourier". In: *C. R. Acad. Sci. Paris* 156 (1913), pp. 1655–1658.

[PR] Jim Portegies and Michael Rothgang. "Formalization of the real interpolation theorem for the Carleson project". Manuscript, in preparation.

[Top22] Adam Topaz. *Definitions in the liquid tensor experiment*. Oct. 14, 2022. URL: https://leanprover-community.github.io/blog/posts/lte-examples/.

Lars Becker, University of Bonn
Email address: becker@math.uni-bonn.de

María Inés de Frutos-Fernández, University of Bonn
Email address: midff@math.uni-bonn.de

Leo Diedering, University of Bonn
Email address: leo.diedering@uni-bonn.de

Floris van Doorn, University of Bonn
Email address: vdoorn@math.uni-bonn.de

Sébastien Gouëzel, University of Rennes
Email address: sebastien.gouezel@univ-rennes1.fr

Evgenia Karunus, University of Bonn
Email address: lakesare@gmail.com

Edward van de Meent, University of Utrecht
Email address: edwardvdmeent@gmail.com

Pietro Monticone, University of Trento
Email address: pit.monticone@gmail.com

Jasper Mulder-Sohn, The Hague
Email address: jasper.mulder@planet.nl

Jim Portegies, Eindhoven University of Technology
Email address: j.w.portegies@tue.nl

Joris Roos, University of Massachusetts Lowell
Email address: joris_roos@uml.edu

Michael Rothgang, University of Bonn
Email address: rothgang@math.uni-bonn.de

James Sundstrom, Baruch College
Email address: james.sundstrom@baruch.cuny.edu

Jeremy Tan, National University of Singapore
Email address: jtjierui@comp.nus.edu.sg