Formalized functional analysis with semilinear maps

Frédéric Dupuis[1*], Robert Y. Lewis[2*] and Heather Macbeth[3*]

[1]Département d'informatique et de recherche opérationnelle, Université de Montréal.
[2]Department of Computer Science, Brown University.
[3]Department of Mathematics, Fordham University.

*Corresponding author(s). E-mail(s): dupuisf@iro.umontreal.ca; robert_lewis@brown.edu; hmacbeth1@fordham.edu;

Abstract

Semilinear maps are a generalization of linear maps between vector spaces where we allow the scalar action to be twisted by a ring homomorphism such as complex conjugation. In particular, this generalization unifies the concepts of linear and conjugate-linear maps. We implement this generalization in Lean's mathlib library, along with a number of important results in functional analysis which previously were impossible to formalize properly. Specifically, we prove the Fréchet–Riesz representation theorem and the spectral theorem for compact self-adjoint operators generically over real and complex Hilbert spaces. We also show that semilinear maps have applications beyond functional analysis by formalizing the one-dimensional case of a theorem of Dieudonné and Manin that classifies the isocrystals over an algebraically closed field with positive characteristic.

Keywords: Functional analysis, Lean, linear algebra, semilinear, Hilbert space

1 Introduction

Proof assistant users have long recognized the value of abstraction. Working at high levels of generality and specializing only when needed can save significant effort in both the long and short term. In program verification, this principle manifests in the use of stepwise refinement of programs from abstract specifications to executable code [1, 2]. Mathematical generalizations that are rarely used in informal presentations are much more common in formal libraries, including the use of filters to generalize limits in topology and analysis [3] and uniform spaces as a generalization of metric spaces [4–6].

We propose another such mathematical generalization: *linear maps*, a fundamental concept in many fields of mathematics, can be seen as a special case of *semilinear maps*. A linear algebra library built on top of this more general structure can unify concepts that would otherwise be defined separately. In particular, linear and *conjugate-linear* (or *antilinear*) maps are both examples of semilinear maps. By relating these, one can avoid a large amount of code duplication and state many theorems more naturally. This generalization is rarely seen explicitly in informal mathematics. Texts tend to focus on the linear case, claiming results about the conjugate-linear or semilinear cases "by analogy" when needed.

Motivated by the desire to formalize theorems from functional analysis at the proper level of abstraction, we have implemented this generalization in mathlib [7], a library of formal mathematics in the Lean proof assistant [8]. When we started this project, much of mathlib was already built on top of standard linear maps. With care and clever notation we were able to make the transition largely invisible. With the generalization complete we were able to state and prove a number of theorems far more elegantly than could have been done before.

Among the results unlocked by this refactor are the Fréchet–Riesz representation theorem, which states that a Hilbert space is either isomorphic or conjugate-isomorphic to its dual space; the generic definition of the adjoint operator on an inner product space over $\mathbb{R}$ or $\mathbb{C}$; and the spectral theorem for compact self-adjoint operators on Hilbert spaces, which gives a canonical form for an important class of linear maps by reference to their eigenvectors. This material in turn lays the groundwork for the formalization of vast areas of mathematics: complex Hilbert spaces are the bread and butter of quantum mechanics and are therefore a prerequisite for quantum information theory and a large part of mathematical physics.

Separately from the theory of semilinear maps, one of the key ingredients for the development of the spectral theorem was a structure theory of Hilbert spaces, notably including Hilbert bases. As a second application of this material, we also developed a theory of Fourier series on the circle, culminating in Parseval's identity.

Finally, as evidence that semilinear maps are useful for more than unifying real and complex vector spaces, we have also formalized the one-dimensional case of a theorem of Dieudonné and Manin [9] that classifies the isocrystals over an algebraically closed field of characteristic $p > 0$. This is a foundational result in p-adic Hodge theory.

Related literature documents the struggles in other libraries to unify real and complex linear algebra. For instance, Aransay and Divasón [10], working in Isabelle, write:

> We miss ... the definition of a "common place" or generic structure representing inner product spaces over real and complex numbers ... that could permit a definition and formalisation of the Gram-Schmidt process for both structures simultaneously.

Their work introduces a "local" solution to the issue, but we argue that basing a library on semilinear maps is the "global" solution. We discuss related work in more detail in Section 11.

We roughly estimate that over the course of this project we have added 17k lines of code to mathlib, with 1k more lines waiting to be merged. We provide links to our contributions, indicating where they can be found in the library, on the project website.[1]

This paper is an expanded version of an earlier conference publication [11]. Section 7.2 contains a development of the theory of Hilbert bases, which we developed concurrently with the Hilbert sum construction (Section 7.1) described in the original article which was used for the spectral theorem. Section 2.5 contains an application of this work on Hilbert bases: an efficient development of the theory of Fourier series, culminating in Parseval's identity. A paragraph at the end of Section 10 describes some additions to the Witt vector development written shortly after the original submission.

2 Mathematical preliminaries

2.1 Semilinear maps

Given modules M_1, M_2 over semirings R_1, R_2 and a ring homomorphism $\sigma : R_1 \to R_2$, a *σ-semilinear map* from M_1 to M_2 is a function $f : M_1 \to M_2$ satisfying the two axioms

1. for all $x, y \in M_1$, $f(x + y) = f(x) + f(y)$
2. for all $x \in M_1$ and $c \in R_1$, $f(cx) = \sigma(c)f(x)$.

Let us note the two canonical examples:

- For $R_1 = R_2 = R$ and σ the identity ring homomorphism $\mathrm{id}_R : R \to R$, the second condition simplifies to $f(cx) = cf(x)$, and therefore an id_R-semilinear map is precisely an R-linear map in the classic sense.
- For $R_1 = R_2 = \mathbb{C}$ and σ the complex-conjugation operation $\mathrm{conj} : \mathbb{C} \to \mathbb{C}$, the second condition simplifies to $f(cx) = \overline{c}f(x)$. Therefore a conj-semilinear map is a conjugate-linear map between complex vector spaces.

The theory of semilinear maps develops along the same lines as the theory of linear maps, with minimal adjustment. The composition of a σ-semilinear map and a τ-semilinear map, for $\sigma : R_1 \to R_2$ and $\tau : R_2 \to R_3$, is a $(\tau \circ \sigma)$-semilinear map. (For example, the composition of two conjugate-linear maps is a linear map.) If σ is bijective, the inverse of a bijective σ-semilinear map is a σ^{-1}-semilinear map.

Theorems about special classes of linear maps also admit semilinear analogues. Consider, for example, the theorem that a $\mathbb{K}$-linear map $f : E_1 \to E_2$, for $\mathbb{K}$ a normed field and E_1, E_2 normed spaces over $\mathbb{K}$, is continuous if and only if it is *bounded* ($\|f(x)\| \le M\|x\|$ for some fixed M, for all x). This theorem generalizes to σ-semilinear maps, for $\sigma : \mathbb{K}_1 \to \mathbb{K}_2$, if the ring homomorphism σ is an isometry.

2.2 Conjugate-linear maps in functional analysis

An *inner product space* is a vector space E over a scalar field $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ equipped with an *inner product* $\langle \cdot, \cdot \rangle$, namely a $\mathbb{K}$-valued function of two arguments which is conjugate-linear in the first argument and linear in the second argument and which has symmetry and positivity properties:

[1] https://robertylewis.com/semilinear-paper

1. for all $u, v, w \in E$, $\langle u + v, w\rangle = \langle u, w\rangle + \langle v, w\rangle$ and $\langle w, u + v\rangle = \langle w, u\rangle + \langle w, v\rangle$;
2. for all $c \in \mathbb{K}$ and $v, w \in E$, $\langle cv, w\rangle = \overline{c}\langle v, w\rangle$ and $\langle v, cw\rangle = c\langle v, w\rangle$;
3. for all $v, w \in E$, $\langle v, w\rangle = \overline{\langle w, v\rangle}$;
4. for all $v \in E$, the quantity $\langle v, v\rangle$ (which by (3) is real) is nonnegative, and strictly positive unless $v = 0$.

For the case of real scalars, $\mathbb{K} = \mathbb{R}$, we consider the conjugation operation as being the identity; this allows a development of the complex case to subsume the simpler real case.

An inner product space has an associated norm $\|v\| = \sqrt{\langle v, v\rangle}$ and hence a metric structure. A *Hilbert space* is an inner product space for which this metric is complete. This condition is automatic in finite dimension.

The *dual* of an inner product space E is the $\mathbb{K}$-vector space of continuous linear maps $\varphi : E \to \mathbb{K}$. There is a natural conjugate-linear map from E to its dual E^*: the vector $v \in E$ is mapped to the vector $\langle v, \cdot\rangle$ in E^*. To see the conjugate-linearity of this map, observe that $\langle cv, \cdot\rangle = \overline{c}\langle v, \cdot\rangle$. It is not difficult to see that, for an appropriate norm on E^*, this map is an isometry. A more subtle theorem, the **Fréchet–Riesz representation theorem**, asserts that for a Hilbert space E this conjugate-linear map is bijective.

Given Hilbert spaces E, F over $\mathbb{K}$ and a continuous linear map $T : E \to F$, it can be proved that there is a unique continuous linear map $T^* : F \to E$, the *adjoint* of T, such that for all $v \in E$ and $w \in F$, $\langle Tv, w\rangle = \langle v, T^*w\rangle$. It turns out that the operation of sending $T : E \to F$ to its adjoint $T^* : F \to E$ is a conjugate-linear map from $E \to F$ to $F \to E$. To see the conjugate-linearity in this case, observe that

$$\langle v, (cT)^*w\rangle = \langle (cT)v, w\rangle = \overline{c}\langle Tv, w\rangle = \overline{c}\langle v, T^*w\rangle = \langle v, (\overline{c}T^*)w\rangle.$$

Like the conjugate-linear map appearing in the Fréchet–Riesz representation theorem, the adjoint map $T \mapsto T^*$ turns out to be bijective and (for an appropriate norm) isometric.

Several important classes of continuous linear maps are defined using the adjoint. A continuous linear map $T : E \to E$ is *self-adjoint*, if $T^* = T$, and it is *normal*, if $T^*T = TT^*$. Self-adjoint implies normal.

2.3 Hilbert sums and Hilbert bases

The *Hilbert sum* $\bigoplus_{i\in\iota} E_\iota$ of a family of inner product spaces $(E_i)_{i\in\iota}$ is an inner product space whose elements are choices $(v_i)_{i\in\iota}$ of an element from each E_i, such that the collection of chosen elements is square-summable in the sense that $\sum_{i\in\iota}\|v_i\|^2 < \infty$. Elements in the Hilbert sum $\bigoplus_{i\in\iota} E_i$ can be added and scalar-multiplied in the obvious way. The inner product on the Hilbert sum is given by $\langle (v_i)_{i\in\iota}, (w_i)_{i\in\iota}\rangle = \sum_{i\in\iota}\langle v_i, w_i\rangle$. It can be proved that if each E_i is a Hilbert space (i.e., complete) then so is $\bigoplus_{i\in\iota} E_i$. A linear map $T : \bigoplus_{i\in\iota} E_i \to \bigoplus_{i\in\iota} E_i$ is *diagonal* if there exist scalars $(\mu_i)_{i\in\iota}$ such that for all $(v_i)_{i\in\iota} \in \bigoplus_{i\in\iota} E_i$, $T((v_i)_{i\in\iota}) = (\mu_i v_i)_{i\in\iota}$.

The Hilbert sum $\bigoplus_{i\in\iota} \mathbb{K}$ of ι copies of the trivial inner product space $\mathbb{K}$ is denoted $\ell^2(\iota, \mathbb{K})$. A *Hilbert basis* for an inner product space E, with the index set ι, is a bijective linear isometry r from E to $\ell^2(\iota, \mathbb{K})$. This is identified, in informal mathematics, with

the collection $(v_i)_{i\in\iota} = (r^{-1}(e_i))_{i\in\iota}$ of vectors in the Hilbert space E which are the preimages of the elementary vectors

$$e_i(j) = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases}$$

in $\ell^2(\iota, \mathbb{K})$: such a collection $(v_i)_{i\in\iota}$ of vectors in E is a Hilbert basis if its elements are mutually-orthogonal and their span in E is dense. Thus the notion generalizes the finite-dimensional notion of an *orthonormal basis*. An argument via Zorn's lemma proves that a Hilbert space admits an orthonormal basis.

The bijective linear isometry $r : E \to \ell^2(\iota, \mathbb{K})$ can be reconstructed from a Hilbert basis considered as family of vectors $(v_i)_{i\in\iota}$ as follows: for w an element of E, the element $r(w)$ of $\ell^2(\iota, \mathbb{K})$ satisfies,

$$r(w)(j) = \langle v_j, w\rangle. \tag{1}$$

Since this map is an isometry,

$$\begin{aligned} \|w\|^2 &= \|r(w)\|^2 \\ &= \sum_{i\in\iota} |\langle v_m, w\rangle|^2 . \end{aligned} \tag{2}$$

2.4 The spectral theorem

A linear map $T : E \to F$ between normed spaces is *compact* if the image under T of the unit ball in E is precompact (that is, has compact closure) in F. This condition implies the continuity of T but is more restrictive. The **spectral theorem** states that a normal (over $\mathbb{C}$) or self-adjoint (over $\mathbb{R}$ or $\mathbb{C}$), compact linear map $T : E \to E$ is equivalent to a diagonal map, in the sense that there exists a bijective linear isometry Φ from E to a Hilbert sum $\bigoplus_{i\in\iota} F_i$, such that the linear map $\Phi \circ T \circ \Phi^{-1}$ is diagonal. In fact, the F_i may be chosen to be the eigenspaces of T, with the μ_i chosen to be the associated eigenvalues.

In finite dimension, every linear map is compact. In this setting the spectral theorem reduces to the more elementary **diagonalization theorem** for a normal endomorphism T of a finite-dimensional inner product space E: there exists a bijective linear isometry Φ from E to a finite direct sum of finite-dimensional inner product spaces $(F_i)_{i\in\iota}$, such that the linear map $\Phi \circ T \circ \Phi^{-1}$ is diagonal.

2.5 Fourier series and Parseval's identity

A concrete example of a complex Hilbert space is $L^2(S^1, \mathbb{C})$, the square-integrable functions on the circle $S^1 := \{z \in \mathbb{C} : |z| = 1\}$, identified up to agreement almost everywhere. The inner product of elements $f, g : S^1 \to \mathbb{C}$ in this Hilbert space is by definition

$$\langle f, g\rangle = \int_{z\in S^1} \overline{f(z)} g(z) d\mu_z,$$

for μ the unit-measure Haar measure on the circle.

The "monomial" functions $F_n : S^1 \to \mathbb{C}$ defined by $F_n(z) = z^n$ are orthonormal in $L^2(S^1, \mathbb{C})$, a straightforward computation, and dense in $L^2(S^1, \mathbb{C})$, a subtle fact which can be proved by invoking the Stone-Weierstrass theorem and the denseness of continuous functions in L^p. Therefore the functions $(F_n)_{n\in\mathbb{Z}}$ form a $\mathbb{Z}$-indexed Hilbert basis for $L^2(S^1, \mathbb{C})$, and define an isometry from $L^2(S^1, \mathbb{C})$ to $\ell^2(\mathbb{Z}, \mathbb{C})$.

The *Fourier coefficients* $\left(\hat{f}_n\right)_{n\in\mathbb{Z}}$ of a function $f \in L^2(S^1, \mathbb{C})$ are by definition the $\mathbb{Z}$-indexed family of complex numbers constituting the element of $\ell^2(\mathbb{Z}, \mathbb{C})$ to which f is mapped under this isometry. Unfolding the definition of the inner product in this Hilbert space, we have by (1) that

$$\hat{f}_n = \int_{z\in S^1} z^{-n} f(z) d\mu_z,$$

and by (2) that

$$\begin{aligned}\int_{z\in S^1} |f(z)|^2 d\mu_z &= \|f\|^2 \\ &= \sum_{n\in\mathbb{Z}} |\hat{f}_n|^2.\end{aligned}$$

This theorem is known as **Parseval's identity**.

2.6 Frobenius-semilinear maps

Given a commutative ring R and a prime p, there is a classical construction [12] of an associated commutative ring $\mathbb{W}(R)$, the ring of *p-typical Witt vectors* of R. The elements of this ring are sequences of elements of R, but the definitions of addition and multiplication are rather elaborate. The motivating example is that for R the finite field $\mathbb{Z}/p\mathbb{Z}$, the ring $\mathbb{W}(\mathbb{Z}/p\mathbb{Z})$ is the ring of p-adic integers.

The ring $\mathbb{W}(R)$ admits a canonical ring-endomorphism, the *Frobenius endomorphism*. Concretely, when R has characteristic p, it operates by sending a sequence $(x_0, x_1, x_2, \ldots)$ to $(x_0^p, x_1^p, x_2^p, \ldots)$. In the example of the p-adic integers $\mathbb{W}(\mathbb{Z}/p\mathbb{Z})$, this endomorphism is the identity, so the construction becomes interesting only for more complicated rings R, such as field extensions of $\mathbb{Z}/p\mathbb{Z}$.

For R an integral domain of characteristic p, the ring $\mathbb{W}(R)$ is also an integral domain, and therefore has a well-defined field of fractions. In this case, the Frobenius endomorphism of $\mathbb{W}(R)$ extends to an endomorphism of its field of fractions. If moreover the ring R is perfect, then the Frobenius endomorphism is an automorphism (that is, bijective), as is the induced automorphism of its field of fractions.

Let us fix an algebraically closed field R of characteristic p (which is necessarily a perfect integral domain), and denote by K the field of fractions of $\mathbb{W}(R)$ and by $\varphi : K \to K$ the Frobenius automorphism of K. There is a very well-developed theory of φ-semilinear maps between vector spaces over K. Notably, an important **theorem of Dieudonné and Manin** [9] provides an analogue of the spectral theorem. For a finite-dimensional vector space V over K, it classifies the *isocrystals* (bijective φ-semilinear maps $f : V \to V$), by constructing for such an f a decomposition of V as a direct sum

of vector spaces V_i which are preserved by f and on each of which the map f has a certain canonical form.

In the one-dimensional case this classification can be stated in a fairly elementary way. Let $f : K \to K$ be a φ-semilinear automorphism of K, considered as a vector space over itself. Then there exists an invertible element a of K and an integer $m \in \mathbb{Z}$, such that for all $v \in K$, $f(v) = p^m a^{-1} \varphi(av)$.

3 Lean preliminaries

The mathlib library builds its algebraic hierarchy using *type classes* [7, 13]. Baanen [14] gives an in depth account of mathlib's use of type classes, which we summarize very briefly.

Each argument to a Lean declaration is declared as *explicit* (`()`), *implicit* (`{}`), or *instance-implicit* (`[]`). Explicit arguments must be provided when the declaration is applied; implicit arguments are inferred by unification; instance-implicit arguments are inferred by type class instance resolution.

The fundamental type class of mathlib's linear algebra library is `module`.

```
class module (R : Type u) (M : Type v) [semiring R] [add_comm_monoid
    M]
  extends distrib_mul_action R M :=
(add_smul : ∀ (r s : R) (x : M), (r + s) · x = r · x + s · x)
(zero_smul : ∀ (x : M), (0 : R) · x = 0)
```

This type class says that the additive monoid `M` has an `R`-module structure: it supports scalar multiplication by elements of the semiring `R`, and this scalar multiplication behaves properly with respect to addition on `M`. When `R` is a field instead of a semiring, an `R`-module is in fact a vector space. Many definitions and theorems apply in the more general setting, and when the vector space setting is needed, the transition is invisible.

A type class is a structure (i.e. a record type) that takes zero or more *parameters* and has zero or more *fields*. In the above, the arguments `R` and `M` are parameters, as are the arguments that `R` is a semiring and `M` is an additive commutative monoid. In order to elaborate the type `module R M`, Lean's type class inference algorithm must be able to infer the latter arguments automatically. The fields of `module` are `add_smul` and `zero_smul`, and a projection to `distrib_mul_action R M`. To construct a term of type `module R M`, the user must provide these values; given a term of type `module R M`, the user can access these values. The `extends` keyword can be read as "inherits from." An assumption `distrib_mul_action R M` is available while defining the fields `add_smul` and `zero_smul`, and indeed, the scalar action used in these fields is derived from this instance.

By default the parameters to a type class are *input parameters*. Lean will begin its instance search when all input parameters are known. By denoting certain parameters as *output parameters*, the user can instruct Lean to begin searching for instances of that class before those parameters are known; they will be determined by the solution to the search. Baanen [14, Section 5.1] describes output parameters in more detail.

Like mathlib, we freely use classical logic and do not focus on defining things computably. Within code blocks in this paper, we omit the bodies of definitions and theorems when only the type is relevant, omit some implicit arguments when the

types are clear from context, and occasionally rename declarations for the sake of presentation.

4 Semilinear maps in Lean

Section 2 covered the mathematical motivation for semilinear maps. Here we focus on our implementation of this generalization in Lean. This work is done in the context of mathlib [7], a project with over 860k lines of code, 240 contributors, and countless users. Given the difficulty and importance of maintaining such a large library [15], we were motivated to make this refactor with as little disruption as possible.

4.1 Defining semilinear maps

Before beginning our refactor to use semilinear maps, mathlib's linear algebra development was based on the more familiar concept of linear maps.

```
structure linear_map (R : Type u) (M₁ : Type v) (M₂ : Type w)
  [semiring R] [add_comm_monoid M₁] [add_comm_monoid M₂] [module R M₁]
  [module R M₂] extends add_hom M₁ M₂, mul_action_hom R M₁ M₂
```

Given two R-modules M_1 and M_2, a linear map is an additive homomorphism $M_1 \to M_2$ that respects the multiplicative action of R. A `mul_action_hom` is a homomorphism between types acted on by the same type of scalars [16].

For readers unfamiliar with Lean syntax, it may be clarifying to see what information goes in to defining such a linear map. Despite the intimidating syntax, the input information is exactly as expected: if you have types R, M_1, and M_2 with the appropriate operations and structure, you can construct a linear map by providing a function $M_1 \to M_2$ and proofs that this function factors through addition and scalar multiplication.

```
example (R : Type u) (M₁ : Type v) (M₂ : Type w)
  [semiring R] [add_comm_monoid M₁] [add_comm_monoid M₂] [module R M₁]
  [module R M₂] : linear_map R M₁ M₂ :=
{ to_fun := _, -- M₁ → M₂
  map_add' := _, -- ∀ (x y : M₁), to_fun (x + y) = to_fun x + to_fun y
  map_smul' := _ } -- ∀ (m : R) (x : M₁), to_fun (m • x) = m • to_fun x
```

As noted in Section 2, the domain and codomain of a linear map are modules over the same semiring. The same is true in the definition of linear equivalences:

```
structure linear_equiv (R : Type u) (M₁ : Type v) (M₂ : Type w)
  [semiring R] [add_comm_monoid M₁] [add_comm_monoid M₂] [module R M₁]
  [module R M₂] extends linear_map R M₁ M₂, add_equiv M₁ M₂
```

The type signature of a semilinear map[2] is more complicated, involving two scalar semirings and a ring homomorphism between them. It no longer makes sense to extend `mul_action_hom`, since the multiplicative actions are over different scalar types, so we instead add the field `map_smul` directly. The arguments R and S can be inferred from σ and are thus marked as implicit. The type `R →+* S` is the type of ring homomorphisms from R to S.

[2] In our mathlib contribution we did not rename the type `linear_map` to `semilinear_map`. This simplified the refactor and makes the definition easier to find for beginners. For the sake of clarity in this paper, we refer to the generalized type by the more accurate name.

```
structure semilinear_map {R : Type*} {S : Type*} [semiring R]
    [semiring S]
  (σ : R →+* S) (M₁ : Type*) (M₂ : Type*)
  [add_comm_monoid M₁] [add_comm_monoid M₂] [module R M₁] [module S M₂]
  extends add_hom M₁ M₂ :=
(map_smul' : ∀ (r : R) (x : M₁), to_fun (r • x) = (σ r) • to_fun x)
```

While the type signature has grown more complicated, the constructor for a semilinear map is quite similar to that of a linear map:

```
example {R : Type*} {S : Type*} [semiring R] [semiring S]
  (σ : R →+* S) (M₁ : Type*) (M₂ : Type*)
  [add_comm_monoid M₁] [add_comm_monoid M₂] [module R M₁] [module S
    M₂] :
  semilinear_map σ M₁ M₂ :=
{ to_fun := _, -- M₁ → M₂
  map_add' := _, -- ∀ (x y : M₁), to_fun (x + y) = to_fun x + to_fun y
  map_smul' := _ } -- ∀ (r : R) (x : M₁), to_fun (r • x) = σ r • to_fun
    x
```

The generalization to semilinear equivalences is similar, but more involved in order to gracefully handle inversion of such maps. The additional parameter σ' and the `ring_hom_inv_pair` type class are explained in Section 4.3.

```
structure semilinear_equiv {R : Type*} {S : Type*} [semiring R]
    [semiring S]
  (σ : R →+* S) {σ' : S →+* R} [ring_hom_inv_pair σ σ']
  [ring_hom_inv_pair σ' σ] (M₁ : Type*) (M₂ : Type*)
  [add_comm_monoid M₁] [add_comm_monoid M₂] [module R M₁] [module S M₂]
  extends linear_map σ M₁ M₂, add_equiv M₁ M₂
```

4.2 Notation for semilinear maps

One can see from these definitions that semilinear maps are not a drop-in replacement for linear maps. The type signature is different, even when looking only at explicit arguments. To convert an `R`-linear map to a semilinear map, one must know to invoke `ring_hom.id R`, the identity ring homomorphism on `R`.

Given how frequently linear maps appear in mathlib, this refactor threatened to be painful. Our job was made immensely easier by the use of notation. Before our refactor mathlib used the notation `M₁ →ₗ[R] M₂` to stand for for `linear_map R M₁ M₂`. By redefining this notation to stand for `semilinear_map (ring_hom.id R) M₁ M₂` we were largely able to avoid breaking definitions and proofs throughout the library. The same approach, with notation `M₁ ≃ₗ[R] M₂`, worked to generalize linear equivalences to semilinear equivalences. We introduced similar notation `M₁ →ₛₗ[σ] M₂` to stand for `semilinear_map σ M₁ M₂`, and `M₁ →ₗ⋆[R] M₂` to stand for a semilinear map with respect to a fixed involution such as complex conjugation.

The composition of linear maps proved to be a complication. As we note in Section 4.3, an additional type class must be inferred to justify that two semilinear maps can be composed. This inference was fragile in the presence of other features, like implicit coercions, that complicate elaboration. We introduced notation $\circ_l$ for

	Linear	Conjugate-linear	Semilinear	Meaning
Map	`M₁ →ₗ[R] M₂`	`M₁ →ₗ⋆[R] M₂`	`M₁ →ₛₗ[σ] M₂`	Between modules; factors over addition and scalar multiplication
Continuous map	`M₁ →L[R] M₂`	`M₁ →L⋆[R] M₂`	`M₁ →SL[σ] M₂`	Between topological modules; a continuous map
Equivalence	`M₁ ≃ₗ[R] M₂`	`M₁ ≃ₗ⋆[R] M₂`	`M₁ ≃ₛₗ[σ] M₂`	An invertible map
Isometry	`M₁ ≃ₗᵢ[R] M₂`	`M₁ ≃ₗᵢ⋆[R] M₂`	`M₁ ≃ₛₗᵢ[σ] M₂`	Between normed modules; a norm-preserving equivalence

Fig. 1 Notation for various classes of (semi)linear operators that appear in this paper

the composition of linear maps, using `ring_hom.id` to justify the composition, and manually inserted this notation where needed.

For our new definition to be useful, theorems stated for linear maps `M₁ →ₗ[R] M₂` needed to be upgraded to theorems about semilinear maps `M₁ →ₛₗ[σ] M₂` when possible. Doing so is mostly mechanical and our use of notation let us approach this without hurry. Because theorems generalized to semilinear maps still apply directly to the linear case we were able to do this generalization incrementally from the bottom up. In particular, several more specialized classes of linear maps and equivalences are also present in mathlib (Figure 1). Our bottom-up approach allowed us to break down the refactor into more manageable pieces by generalizing these one at a time.

4.3 Composition of semilinear maps

Composition of maps is complicated by this generalization. The composition of two linear maps is straightforward: it is easy to check that the composition of the underlying functions preserves addition and scalar multiplication. With semilinear maps one must also compose the homomorphisms between scalar rings. Given `f : M₁ →ₛₗ[σ₁₂] M₂` and `g : M₂ →ₛₗ[σ₂₃] M₃`, we would naturally end up with `g.comp f : M₁ →ₛₗ[σ₂₃.comp σ₁₂] M₃`.

This ends up being awkward to handle in many common situations. Suppose we wish to state that `f : M₁ →ₛₗ[σ₁₂] M₂` and `g : M₂ →ₛₗ[σ₂₁] M₁` are inverse maps: `f.comp g = (id : M₁ →ₗ[R] M₁)`. This statement is not type-correct, since the ring homomorphism on the left is `σ₁₂.comp σ₂₁` and the one on the right is the identity. Such an issue appears in practice, for example, when defining the adjoint as a conjugate-linear map (Section 6).

To solve this issue, we introduce a type class `ring_hom_comp_triple` that states that two ring homomorphisms compose to a third.

```
class ring_hom_comp_triple [semiring R₁] [semiring R₂] [semiring R₃]
  (σ₁₂ : R₁ →+* R₂) (σ₂₃ : R₂ →+* R₃) (σ₁₃ : out_param (R₁ →+* R₃)) :
    Prop :=
(comp_eq : σ₂₃.comp σ₁₂ = σ₁₃)
```

We register a number of global instances of this class. We then use the `ring_hom_comp_ triple` type class in the definition of composition.

```
def semilinear_map.comp {R₁ R₂ R₃ : Type*} {M₁ M₂ M₃ : Type*}
  [semiring R₁] [semiring R₂] [semiring R₃]
  [add_comm_monoid M₁] [add_comm_monoid M₂] [add_comm_monoid M₃]
  {mod_M₁ : module R₁ M₁} {mod_M₂ : module R₂ M₂} {mod_M₃ : module R₃
    M₃}
  {σ₁₂ : R₁ →+* R₂} {σ₂₃ : R₂ →+* R₃} {σ₁₃ : R₁ →+* R₃}
  [ring_hom_comp_triple σ₁₂ σ₂₃ σ₁₃]
  (g : M₂ →ₛₗ[σ₂₃] M₃) (f : M₁ →ₛₗ[σ₁₂] M₂) :
  (M₁ →ₛₗ[σ₁₃] M₃)
```

While this may appear to be a rather verbose type signature for the composition of maps, it allows us to avoid the above problem without introducing further complications. In common situations, the appropriate global instances generate the necessary `ring_hom_comp_triple` argument without input from the user. For example, the following global instance allows for the composition of two (genuine) linear maps, or more generally for the composition of a semilinear map with a linear map.

```
instance [semiring R₁] [semiring R₂] {σ₁₂ : R₁ →+* R₂} :
  ring_hom_comp_triple (ring_hom.id R₁) σ₁₂ σ₁₂
```

Another instance helps in the setting of conjugate-linear maps.[3]

```
instance [comm_semiring R] [star_ring R] :
  ring_hom_comp_triple (conj R) (conj R) (ring_hom.id R)
```

We expand on the types here in Section 5.1; in concrete terms, this instance says that the conjugation operation on a type supporting conjugation is an involution. This allows us to compose two conjugate-linear maps to obtain, definitionally, a linear map. The intention is that users should never work directly with a composition `g.comp f : M₁ →ₛₗ[σ₂₃.comp σ₁₂] M₃`, but instead with `g.comp f : M₁ →ₛₗ[σ₁₃] M₃` for some σ_{13} satisfying `ring_hom_comp_triple σ₁₂ σ₂₃ σ₁₃`, which is strictly more general.

Similar issues appear with semilinear equivalences, specifically when defining the symmetric equivalence: if `e : E ≃ₛₗ[σ] F`, the "natural" definition of the symmetric equivalence would give `e.symm : F ≃ₛₗ[σ.symm] E`. Some ring homomorphisms, notably conjugation on $\mathbb{C}$, have the property that σ`.symm = `σ. But these equalities are rarely definitional and spurious `symm`s can block type checking. Introducing a new type class `ring_hom_inv_pair` that states that two ring homomorphisms are inverses of each other, analogous to the type class `ring_hom_comp_triple` described above, again solves this issue.

[3] In fact, we do not state this instance explicitly; it is derived by type class inference from the `ring_hom_inv_pair` instance for `conj` (see below) and yet another global instance generating a `ring_hom_comp_triple` with the identity from a `ring_hom_inv_pair`.

```
class ring_hom_inv_pair [semiring R₁] [semiring R₂] (σ : R₁ →+* R₂)
  (τ : out_param (R₂ →+* R₁)) : Prop :=
(comp_eq : τ.comp σ = ring_hom.id R₁)
(comp_eq₂ : σ.comp τ = ring_hom.id R₂)
```

Now, with a suitable instance stating that the conjugation operation on a type supporting conjugation is its own inverse, we can work with a conjugate-linear equivalence `e : E ≃ₗ⋆[R] F`, i.e. `e : E ≃ₛₗ[conj] F`, over scalars of that type, and have that its inverse `e.symm` be genuinely of type `F ≃ₗ⋆[R] E`.

```
instance [comm_semiring R] [star_ring R] : ring_hom_inv_pair (conj R)
    (conj R)
```

5 Fréchet–Riesz representation theorem

In the following three sections we describe results that we were able to formalize at the proper level of generality thanks to our refactor. By the "proper level" of generality, we mean that our results hold generically over the real and complex numbers without case splits or separate declarations.

5.1 The `is_R_or_C` type class

Many results in functional analysis, including those presented here, hold for a field $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. Such results are usually presented in the literature by giving proofs for the complex case, with the real case following in the obvious way: replace complex conjugation by the identity, i by zero, and so on.

Before beginning our refactor, we introduced a type class `is_R_or_C` to mathlib used to formalize this kind of result. A type that instantiates `is_R_or_C` is a complete nondiscrete field with (real) norm containing an element i and functions `conj`, `re` and `im` that satisfy a number of ad-hoc axioms chosen to mimic the behavior of a field that is either $\mathbb{R}$ or $\mathbb{C}$. The `conj` operator is an involutive ring homomorphism, enabling the notation discussed in Section 4.3. Two global instances stating `is_R_or_C ℝ` and `is_R_or_C ℂ` allow theorems over the generic type class to be specialized immediately to either concrete type. The conjugation operator `conj` is definitionally equal to the identity function in the real case and the complex conjugation function in the complex case. We note an experiment with a similar type class in Isabelle [10].

Working over an `is_R_or_C` field enables many nice features. In particular, the conjugation operator `conj` is definitionally equal to the identity function in the real case and the complex conjugate in the complex case. Hilbert spaces in mathlib are defined over `is_R_or_C` fields. Given two Hilbert spaces `E` and `F` over a field `K`, conjugate-linear maps `E ≃ₗ⋆[K] F` are precisely maps which are semilinear with respect to `conj`, and thus in the real case are linear maps by definition. Within mathlib, this type class has already been used extensively beyond the results mentioned in this paper, notably by Sébastien Gouëzel for stating in correct generality the Hahn–Banach theorem, the smooth case of the inverse function theorem, and more.

```
local notation `𝒦` := algebra_map ℝ _

class is_R_or_C (K : Type*) extends nondiscrete_normed_field K,
  star_ring K, normed_algebra ℝ K, complete_space K :=
(re : K →+ ℝ)
(im : K →+ ℝ)
(I : K)
(I_re_ax : re I = 0)
(I_mul_I_ax : I = 0 ∨ I * I = -1)
(re_add_im_ax : ∀ (z : K), 𝒦 (re z) + 𝒦 (im z) * I = z)
(of_real_re_ax : ∀ r : ℝ, re (𝒦 r) = r)
(of_real_im_ax : ∀ r : ℝ, im (𝒦 r) = 0)
(mul_re_ax : ∀ z w : K, re (z * w) = re z * re w - im z * im w)
(mul_im_ax : ∀ z w : K, im (z * w) = re z * im w + im z * re w)
(conj_re_ax : ∀ z : K, re (conj z) = re z)
(conj_im_ax : ∀ z : K, im (conj z) = -(im z))
(conj_I_ax : conj I = -I)
(norm_sq_eq_def_ax : ∀ (z : K), ‖z‖^2 = (re z) * (re z) + (im z) * (im
    z))
(mul_im_I_ax : ∀ (z : K), (im z) * im I = im z)
(inv_def_ax : ∀ (z : K), z⁻¹ = conj z * 𝒦 ((‖z‖^2)⁻¹))
(div_I_ax : ∀ (z : K), z / I = -(z * I))
```

Fig. 2 The `is_R_or_C` type class is satisfied only by fields isomorphic to $\mathbb{R}$ or $\mathbb{C}$. The `star_ring` assumption endows `K` with an involutive operator `conj` that respects addition and multiplication.

5.2 Fréchet–Riesz representation theorem

Our first application of semilinear maps is in proving the Fréchet–Riesz representation theorem. While the real case has been formalized in Coq [17] and Mizar [18], and the complex case in Isabelle [19], we are not aware of a development that unifies the two.[4]

Given a Hilbert space E, its *dual space* E^* consists of the set of continuous linear functionals on E (i.e. $E^* = \{f : E \to \mathbb{K} \mid f \text{ is linear and continuous}\}$). The dual space certainly includes elements of the form f_v that map $w \in E$ to $\langle v, w\rangle$, and the Fréchet–Riesz representation theorem states that all elements of the dual space are of this form. That is, there exists an (in fact, isometric) equivalence between E and E^* that maps v to to f_v.

The difficulty in formalizing this is that while this equivalence is linear in the real case, in the complex case, it is *conjugate*-linear. The challenge is to construct this object in such a way that (1) there is a common definition for both the real and complex case, and (2) the added complication of conjugate-linearity is completely transparent in the real case. Before our refactor mathlib simply had two separate constructions. We are able to replace those two constructions with the following, which satisfies both requirements stated above:

```
def to_dual [is_R_or_C 𝕂] [inner_product_space 𝕂 E] [complete_space
    E] :
```

[4] This theorem should not be confused with the Riesz–Markov–Kakutani representation theorem, which has been formalized in Mizar, PVS (unfortunately referred to as the "Riesz representation theorem"), and possibly other proof assistants.

```
  E ≃ₗᵢ⋆[𝕜] normed_space.dual 𝕜 E

lemma to_dual_apply [is_R_or_C 𝕜] [inner_product_space 𝕜 E]
  [complete_space E] {x y : E} : to_dual 𝕜 E x y = ⟪x, y⟫
```

Read aloud this definition says that "a real or complex Hilbert space E is isometrically conjugate-isomorphic to its dual space." But when specialized to the real case, the statement is definitionally equal to "E is isometrically isomorphic to its dual space."

Our proof of this theorem does not differ from the real version of the proof in mathlib prior to our refactor. In fact, the patch unifying the real and complex versions[5] added only 45 lines of code and removed 79; the only change beyond rearranging and documentation was to generalize the statement of the theorem. The Lean implementation of the orthogonal projection on real inner product spaces, a tool used in the proof, had been written by Zhouhang Zhou as a port of work in Coq by Boldo et al. [17].

6 Adjoints of operators on Hilbert spaces

Given a continuous linear map A between two Hilbert spaces E and F, the adjoint of A is the unique continuous linear map $A^* : F \to E$ such that for all $x \in E$ and $y \in F$, $\langle y, Ax\rangle_F = \langle A^* y, x\rangle_E$. The adjoint satisfies a number of properties: it is involutive (i.e. $(A^*)^* = A$), it is an isometry, and, most importantly for our purposes here, it is conjugate-linear. Hence, it was natural to bundle it in mathlib as a conjugate-linear isometric equivalence as follows:

```
def continuous_linear_map.adjoint [is_R_or_C 𝕜] [inner_product_space 𝕜
    E]
  [inner_product_space 𝕜 F] [complete_space E] [complete_space F] :
  (E →L[𝕜] F) ≃ₗᵢ⋆[𝕜] (F →L[𝕜] E)

lemma continuous_linear_map.adjoint_inner_left [is_R_or_C 𝕜]
  [inner_product_space 𝕜 E] [inner_product_space 𝕜 F] [complete_space
    E]
  [complete_space F] (A : E →L[𝕜] F) (x : E) (y : F) :
  ⟪continuous_linear_map.adjoint A y, x⟫ = ⟪y, A x⟫
```

This definition fully exploits the algebraic formalism built for semilinear maps, including the composition mechanism of Section 4.3. For example, the statement that the composition of the adjoint operation with itself is equal to the identity map from `E →L[𝕜] F` to itself (a "true" $\mathbb{K}$-linear map) would not typecheck without the `ring_hom_comp_triple` mechanism.

In finite dimension, every linear map is a continuous linear map, so the adjoint construction actually applies to every linear map. We provide this construction as `linear_map.adjoint` for the benefit of future users interested only in the finite-dimensional setting.

An operator T on a Hilbert space is said to be *self-adjoint* if $T = T^*$ and *normal* if $T^*T = TT^*$. We allow these definitions to apply both to the finite-dimensional setting with `linear_map.adjoint` and to the general setting with `continuous_linear_map.adjoint` by in fact writing these definitions in the more

[5] https://github.com/leanprover-community/mathlib/pull/9924

general context of a `star_ring`, a ring equipped with a fixed involutive ring homomorphism.

```
def self_adjoint [ring R] [star_ring R] : add_subgroup R :=
{ carrier := {x | star x = x},   ... }

def is_star_normal [ring R] [star_ring R] (x : R) :=
star x * x = x * star x
```

When `R` is the ring `E →ₗ[𝕂] E` of linear endomorphisms of a finite-dimensional inner product space `E` (with ring operation composition), the involution `star` is `linear_map.adjoint`. When `R` is the ring `E →L[𝕂] E` of continuous linear endomorphisms of a Hilbert space `E`, the involution `star` is `continuous_linear_map.adjoint`.

7 Structure theory of Hilbert spaces

7.1 The dependent-ℓ^p construction

The spectral theorem, in finite dimension also known as the diagonalization theorem, expresses an operator on a Hilbert space in the canonical form of a "diagonal" operator. To describe this canonical form, one needs some version of the Hilbert sum construction, described in this section.

Before we started, mathlib already had a finitary version of this construction, namely a type `pi_Lp` denoting the product of finitely many normed spaces (note that the `p` parameter is not actually used in its definition):

```
def pi_Lp {ι : Type u} (p : ℝ) (G : ι → Type v) : Type (max u v) :=
Π (i : ι), G i
```

as well as a normed space structure that depends on `p` on this product, and, in the case $p = 2$, an inner product space structure:

```
instance pi_Lp.inner_product_space {𝕂 : Type w} [is_R_or_C 𝕂]
  {ι : Type u} [fintype ι] (G : ι → Type v)
  [Π (i : ι), inner_product_space 𝕂 (G i)] :
  inner_product_space 𝕂 (pi_Lp 2 G)
```

We require the general version of this construction, with a possibly-infinite index set ι. We first define a predicate `mem_ℓp f p` on dependent functions in `Π (i : ι), G i` and extended nonnegative real numbers `p`, which, for $p = 2$, amounts to the norm-squared of the function being a convergent sum. The associated subset of `Π (i : ι), G i` is named `lp G p`:

```
def lp {ι : Type u} (G : ι → Type v) [Π (i : ι), normed_group (G i)]
  (p : ℝ≥0∞) : add_subgroup (Π (i : ι), G i) :=
{ carrier := {f | mem_ℓp f p},   ... }
```

In the case $p = 2$ this inner product space is known as the *Hilbert sum* of the family `G`. For general p, in the special case of the trivial family `λ (i : ι), 𝕂` of normed spaces, this construction gives what is traditionally called $\ell^p(\iota, \mathbb{K})$. We will refer to the general construction `lp` as the *dependent-ℓ^p construction*.

We prove `lp G p` to be an additive subgroup, and for $p = 2$ equip it with an inner product space structure.

```
instance lp.inner_product_space {ι : Type u} {𝕂 : Type w} [is_R_or_C 𝕂
    ]
  {G : ι → Type v} [Π (i : ι), inner_product_space 𝕂 (G i)] :
  inner_product_space 𝕂 (lp G 2)
```

This is a reasonably labor-intensive construction (some 500 lines of code), the difficulties being a series of small analytic arguments about the convergence of the sums involved. It is closely analogous to Rémy Degenne's mathlib construction (see also related work in Isabelle [20]) of the normed space structure on $L^p(X, G)$, a subtype of the type of equivalence classes of measurable functions from a measure space X into a normed space G, which we also need in this work (see Section 9.2). However, neither construction is a strict generalization of the other: the L^p construction allows for integrals with respect to an arbitrary measure rather than just sums, whereas the dependent-ℓ^p construction applies to dependent functions of type `Π (i : ι), G i` in which the "codomain" varies depending on the argument. We in fact need this dependent property for the spectral theorem.

A further analytic argument establishes the completeness of `lp`. The key step here is an argument that a pointwise limit of a uniformly-bounded sequence of elements of `lp` is itself in `lp`. is A Hilbert space is by definition a complete inner product space and therefore this establishes that the Hilbert sum `lp G 2` is a Hilbert space.

```
instance lp.complete_space {ι : Type u} {G : ι → Type v}
  [Π (i : ι), normed_group (G i)] [∀ (i : ι), complete_space (G i)]
  {p : ℝ≥0∞} [fact (1 ≤ p)] : complete_space (lp G p)
```

Finally, given a Hilbert space E of interest, an important argument establishes a mechanism for "collating" a family of isometries from the summands `G i` into E to an isometric isomorphism from `lp G 2` into E. It is sufficient (and necessary) that the images of the family of isometries form a mutually-orthogonal family of subspaces of E, and that their joint span be dense in E.

```
def orthogonal_family.linear_isometry_equiv [is_R_or_C 𝕂]
  [inner_product_space 𝕂 E] [complete_space E]
  [Π (i : ι), inner_product_space 𝕂 (G i)] {V : Π (i : ι), G i →ₗᵢ[𝕂]
    E}
  (hV : orthogonal_family 𝕂 V) [∀ (i : ι), complete_space (G i)]
  (hV' : (⨆ (i : ι), (V i).to_linear_map.range).topological_closure = ⊤
    ) :
  E ≃ₗᵢ[𝕂] (lp G 2)
```

We also provide the finitary, i.e. `pi_Lp`, version of this construction.

```
def direct_sum.submodule_is_internal.isometry_L2_of_orthogonal_family
  [is_R_or_C 𝕂] [inner_product_space 𝕂 E] [fintype ι] [decidable_eq ι]
  {V : ι → submodule 𝕂 E} (hV : direct_sum.submodule_is_internal V)
  (hV' : orthogonal_family 𝕂 (λ (i : ι), (V i).subtypeₗᵢ)) :
  E ≃ₗᵢ[𝕂] pi_Lp 2 (λ (i : ι), V i)
```

7.2 Hilbert bases

The current definition of bases in mathlib, due to Reid Barton, Mario Carneiro and Anne Baanen, dates back to 2021. Given a vector space E over R, a basis for E is

given by an isomorphism between E and finitely-supported functions from an index type ι to R. This is implemented via the structure

```
structure basis (ι R E : Type*) [semiring R] [add_comm_monoid E]
  [module R E] :=
(repr : E ≃ₗ[R] (ι →₀ R))
```

Note that this looks somewhat different from the traditional definition as a set of linearly-independent vectors that span the space.

Our treatment of Hilbert bases is analogous: rather than directly following the traditional definition as an orthogonal set in a Hilbert space with a dense span, we instead choose to define a Hilbert space version of `repr` in an appropriate way. More precisely, we replace the isomorphism between E and finitely-supported functions from ι to R by a bijective linear isometry between E and $\ell^2(\iota, \mathbb{K})$ (this is the traditional ℓ^2, i.e., the Hilbert sum of ι copies of $\mathbb{K}$, for which we locally introduce the notation `ℓ²(ι, 𝕜)` in mathlib). Moving to an isometry allows us to preserve the inner product, and we include infinite sums by replacing finitely-supported functions by square-summable functions from ι to $\mathbb{K}$.

The result looks like the following. We define a *Hilbert basis* of a Hilbert space E to be a structure whose single field `hilbert_basis.repr` is an isometric isomorphism of E with `ℓ²(ι, 𝕜)`. This parallels the definition of "basis", as an isomorphism of an R-module with `ι →₀ R`.

```
structure hilbert_basis (ι 𝕜 E : Type*) [is_R_or_C 𝕜]
  [inner_product_space 𝕜 E] :=
(repr : E ≃ₗᵢ[𝕜] ℓ²(ι, 𝕜))
```

We can then recover the traditional interpretation as a family of vectors as an instance of the function-coercion (`has_coe_to_fun`) typeclass:

```
instance [is_R_or_C 𝕜] [inner_product_space 𝕜 E] :
  has_coe_to_fun (hilbert_basis ι 𝕜 E) (λ _, ι → E) :=
{ coe := λ b i, b.repr.symm (lp.single 2 i (1:𝕜)) }
```

This allows us to treat `b : hilbert_basis ι 𝕜 E` as a function from ι to E, such that `b i` corresponds to the i-th basis vector. These are orthogonal and have dense span.

We then get the following fundamental property of Hilbert bases:

```
lemma hilbert_basis.repr_apply_apply [is_R_or_C 𝕜]
    [inner_product_space 𝕜 E]
  (b : hilbert_basis ι 𝕜 E) (v : E) (i : ι) :
  b.repr v i = ⟪b i, v⟫
```

The main point of Hilbert bases is to be able to represent any vector in the space as an infinite linear combination of vectors from the basis. Here is what this fact looks like in our treatment of Hilbert bases:

```
lemma hilbert_basis.has_sum_repr [is_R_or_C 𝕜] [inner_product_space 𝕜
    E]
  (b : hilbert_basis ι 𝕜 E) (x : E) :
  has_sum (λ i, b.repr x i • b i) x
```

To construct a Hilbert basis from a traditional representation as an orthonormal family of vectors whose span is dense in the whole module is a Hilbert basis, we can use the following definition:

```
def hilbert_basis.mk [is_R_or_C 𝕂] [inner_product_space 𝕂 E]
  [complete_space E] {v : ι → E} (hv : orthonormal 𝕂 v)
  (hsp : (span 𝕂 (set.range v)).topological_closure = ⊤) :
  hilbert_basis ι 𝕂 E
```

Finally, we also proved the fact that every Hilbert space admits a Hilbert basis, which uses Zorn's lemma:

```
lemma exists_hilbert_basis (𝕂 E : Type*) [is_R_or_C 𝕂]
  [inner_product_space 𝕂 E] [complete_space E] :
  ∃ (w : set E) (b : hilbert_basis w 𝕂 E), (b : w → E) = (coe : w →
    E)
```

Notice that paper versions of this statement often require separability of the Hilbert space, since they additionally stipulate that a Hilbert basis must be countable. This is not the case for us, and in the non-separable case, we simply get an uncountable index type ι.

8 Versions of the spectral theorem

8.1 Common outline of the spectral theorems

A diagonal operator on `lp G 2` or `pi_Lp 2 G` is an operator that, for some fixed sequence of scalars μ : $\iota \to \mathbb{K}$, sends each dependent function `f : Π (i : ι), G i` to the pointwise-rescaled function `λ i, μ i · f i`. The spectral theorem for compact self-adjoint (respectively, normal) operators states that such an operator over `is_R_or_C` (respectively, $\mathbb{C}$) is equivalent to a diagonal operator on `lp G 2`, for some family of inner product spaces G. The finite-dimensional special case, the diagonalization theorem, states that a normal endomorphism of a finite-dimensional inner product space over $\mathbb{C}$ is equivalent to a diagonal operator on some `pi_Lp 2 G`.

The key point of all such theorems, which we defer discussing to Section 8.2, is a proof that every operator from the stated class has an eigenvalue (unless the operator is the trivial operator on the trivial vector space). The proof of this important point is what differs from theorem to theorem. In this subsection we discuss the common part of the proofs of the theorems, namely the reduction to the existence of an eigenvalue.

This part is essentially algebraic and is carried out for a endomorphism of an inner product space E that satisfies the following property, common to those three cases:

```
def inner_product_space.is_normal (T : E →ₗ[𝕂] E) : Prop :=
  ∃ (T' : E →ₗ[𝕂] E), T' * T = T * T' ∧ ∀ x y, ⟪T' x, y⟫ = ⟪x, T y⟫
```

We first show that the eigenspaces of such an operator are mutually orthogonal.

```
lemma orthogonal_family_eigenspaces [is_R_or_C 𝕂]
    [inner_product_space 𝕂 E]
  {T : E →ₗ[𝕂] E} (hT : inner_product_space.is_normal T) :
  orthogonal_family 𝕂 (λ (μ : 𝕂), (eigenspace T μ).subtypeₗᵢ)
```

This puts us in a position to apply the final construction from Section 7.1 to the collection of eigenspaces of T. Specifically, if the completeness property `(⨆ (μ : 𝕂), (eigenspace T μ)).topological_closure = ⊤` or its finite-dimensional analogue can be established, then those results establish an isometric isomorphism between E

and the Hilbert sum of its own eigenspaces. It is easy to check that the operator T, when transferred by this isometric isomorphism to the Hilbert sum, is diagonal.

A further sequence of lemmas leads to this completeness property, and it is here that the eigenvalue existence result is required. It is shown that an `inner_product_space.is_normal` operator preserves orthogonal complements of eigenspaces.

```
lemma invariant_orthogonal_eigenspace [is_R_or_C 𝕂]
    [inner_product_space 𝕂 E]
  {T : E →ₗ[𝕂] E} (hT : inner_product_space.is_normal T) (μ : 𝕂) (v :
     E)
  (hv : v ∈ (eigenspace T μ)⊥) :
  T v ∈ (eigenspace T μ)⊥
```

Such an operator preserves the mutual orthogonal complement of all its eigenspaces.

```
lemma orthogonal_supr_eigenspaces_invariant [is_R_or_C 𝕂]
  [inner_product_space 𝕂 E] {T : E →ₗ[𝕂] E}
  (hT : inner_product_space.is_normal T) {|v : E|}
  (hv : v ∈ (⨆ (μ : 𝕂), eigenspace T μ)⊥) :
  T v ∈ (⨆ (μ : 𝕂), eigenspace T μ)⊥
```

The restriction of such an operator to this mutual orthogonal complement, which is therefore well-defined, itself has no eigenvalues.

```
lemma orthogonal_supr_eigenspaces [is_R_or_C 𝕂] [inner_product_space 𝕂
     E]
  {T : E →ₗ[𝕂] E} (hT : inner_product_space.is_normal T) (μ : 𝕂) :
  eigenspace (T.restrict (orthogonal_supr_eigenspaces_invariant hT)) μ
     = ⊥
```

From here, if the existence of an eigenvalue for all nontrivial operators in the class considered is known, by contraposition the subspace `(⨆ (μ : 𝕂), eigenspace T μ)⊥` (being the domain of the operator `T.restrict (orthogonal_supr_eigenspaces_invariant hT)`, which has no eigenvalues) must be trivial. Standard Hilbert space theory implies that the subspace `⨆ (μ : 𝕂), eigenspace T μ` must be dense, the desired completeness result.

8.2 Existence of an eigenvalue

The first version of the spectral theorem we prove is for normal endomorphisms of a finite-dimensional inner product space over $\mathbb{C}$.

```
def diagonalization [inner_product_space ℂ E] [finite_dimensional ℂ E]
  {T : E →ₗ[ℂ] E} (hT : is_star_normal T) :
  E ≃ₗᵢ[ℂ] pi_Lp 2 (λ μ : eigenvalues T, eigenspace T μ)

lemma diagonalization_apply_self_apply [inner_product_space ℂ E]
  [finite_dimensional ℂ E] {T : E →ₗ[ℂ] E} (hT : is_star_normal T) (v
    : E)
  (μ : eigenvalues T) :
  diagonalization hT (T v) μ = (μ : ℂ) · (diagonalization hT) v μ
```

We also provide the more classical version of this theorem, stating that there exists an orthonormal basis of eigenvectors of `T`.

For this class of operators, the proof of the existence of an eigenvalue is straightforward. In finite dimension, an endomorphism has a well-defined characteristic polynomial. Over an algebraically closed field this polynomial must have a root, and this root is an eigenvalue.

The second version of the spectral theorem we prove is for self-adjoint compact operators on a Hilbert space. Here a map between normed spaces is said to be compact, if the image of every bounded subset has compact closure.

```
def compact_map [nondiscrete_normed_field 𝕂] [normed_group E]
  [normed_space 𝕂 E] [normed_group F] (T : E → F) : Prop :=
∀ s : set E, metric.bounded s → is_compact (closure (T '' s))
```

A compact linear map is automatically continuous, so it is no loss of generality to take `T` to be of type `E →L[𝕂] E`. In this setting we state the spectral theorem as follows.

```
def diagonalization' [is_R_or_C 𝕂] [inner_product_space 𝕂 E]
  [complete_space E] {T : E →L[𝕂] E} (hT : T ∈ self_adjoint (E →L[𝕂]
    E))
  (hT_cpct : compact_map T) :
  E ≃ₗᵢ[𝕂] (lp (λ μ, eigenspace (T : E →ₗ[𝕂] E) μ) 2)

lemma diagonalization_apply_self_apply' [is_R_or_C 𝕂]
  [inner_product_space 𝕂 E] [complete_space E] {T : E →L[𝕂] E}
  (hT : T ∈ self_adjoint (E →L[𝕂] E)) (hT_cpct : compact_map T) (v :
    E)
  (μ : 𝕂) :
  diagonalization' hT hT_cpct (T v) μ = μ · diagonalization' hT hT_cpct
    v μ
```

For this class of operators, the proof of the existence of an eigenvalue comes from a long and delicate calculation involving the Rayleigh quotient, some 700 lines of code. It is proved that local maxima/minima of the Rayleigh quotient are eigenvectors, that the operator norm of `T` is the supremum of the absolute value of the Rayleigh quotient, and (using the compactness of `T`) that the Rayleigh quotient of `T` achieves its maximum.

Having established in this project the basic properties of compact operators, the infinite-dimensional theorem of the spectral theorem for compact normal operators is also within reach. There, the proof of the existence of an eigenvalue comes from an argument about the resolvent, a holomorphic function with values in the Banach space `E →ₗ[ℂ] E`. The current development of complex analysis in mathlib by Yury Kudryashov [21] is sufficiently general for this setting. However, this would not supersede the spectral theorem we prove for compact self-adjoint operators: the latter works generically over $\mathbb{R}$ and $\mathbb{C}$, which is more elegant than to deduce it in the real setting from the normal-operator version over $\mathbb{C}$ by making an argument about the operator's complexification.

9 Fourier series and Parseval's identity

9.1 Continuous functions are dense in L^p

This section describes a technical theorem in measure theory which is a preliminary for the work on Fourier series in Subsection 9.2.

The theorem states that for a normal topological space α equipped with a weakly regular Borel measure μ and a real normed space E, and for $1 \le p < \infty$, the bounded continuous functions are dense in $L^p(\alpha, E)$. Note that for $p = \infty$ this result is not true: the characteristic function of the interval $[0, \infty)$ cannot be continuously approximated in $L^\infty(\mathbb{R}, \mathbb{R})$.

In Lean we state this theorem by introducing the additive subgroup `Lp.bounded_continuous_function` of $L^p(\alpha, E)$ consisting of equivalence classes containing a bounded continuous representative, and stating that the topological closure of this subgroup is equal to the "maximal" subgroup, E itself.

```
lemma bounded_continuous_function_dense [measurable_space α]
  [topological_space α] [normal_space α] [borel_space α]
    [normed_group E]
  [second_countable_topology_either α E] [normed_space ℝ E] {p : ℝ≥
    0∞}
  [fact (1 ≤ p)] (hp : p ≠ ∞) (μ : measure α) [μ.weakly_regular] :
  (bounded_continuous_function E p μ).topological_closure = ⊤
```

The proof is in three steps. First, it suffices to prove the result for a scalar multiple of a characteristic function of a measurable set s. This is because simple functions are dense in L^p, a result we obtained by refactoring an existing result in mathlib (contributed by Zhouhang Zhou) for L^1 functions. Here is the most convenient form of that result, phrased as an induction principle: to prove something for an arbitrary L^p function in a second countable Borel normed group, it suffices to show that

- the property holds for (multiples of) characteristic functions;
- is closed under addition;
- the set of functions in L^p for which the property holds is closed.

```
lemma Lp.induction [measurable_space α] [normed_add_comm_group E] {p :
     ℝ≥0∞}
  [fact (1 ≤ p)] (hp_ne_top : p ≠ ∞) {μ : measure α} (P : Lp E p μ →
    Prop)
  (h_ind : ∀ (c : E) {s : set α} (hs : measurable_set s) (hμs : μ s <
    ∞),
      P (Lp.simple_func.indicator_const p hs hμs.ne c))
  (h_add : ∀ {f g}, ∀ hf : mem_Lp f p μ, ∀ hg : mem_Lp g p μ,
    disjoint (support f) (support g) → P (hf.to_Lp f) → P (hg.to_Lp
    g)
    → P ((hf.to_Lp f) + (hg.to_Lp g)))
  (h_closed : is_closed {f : Lp E p μ | P f}) :
  ∀ f : Lp E p μ, P f
```

The second step in the denseness result for continuous functions is to approximate the given measurable set s above by an open set and below by a closed set; this is a consequence of the weak regularity of the measure μ. The third and final step is to

find a continuous function interpolating between these two sets. This is a consequence of Urysohn's lemma, contributed to mathlib by Yury Kudryashov, and is the step at which we use the fact that the domain α is normal.

We also establish several variants of this theorem. The version used for the Fourier series construction is that for compact α and finite-measure μ, the natural continuous linear map from $C(\alpha, E)$ to $L^p(\alpha, E)$ has dense range.

9.2 Construction of an explicit Hilbert basis for $L^2(S^1, \mathbb{C})$

Fourier series are usually considered for functions on a finite interval such as $[0, 1]$ or $[0, 2\pi]$. In this point of view, the Fourier coefficients of a function $f : [0, 2\pi] \to \mathbb{C}$ are its integrals against the functions $\left(e^{int}\right)_{n\in\mathbb{Z}}$ for $t \in [0, 2\pi]$.

In this work we take an approach which requires less special-case analysis, and which is suggestive of the more general theory of the Pontrjagin dual of a topological group: to consider Fourier series for functions on the unit circle $S^1 := \{z \in \mathbb{C} : |z| = 1\}$, in Lean `circle`. In this point of view, the Fourier coefficients are obtained as the integrals of a function $f : S^1 \to \mathbb{C}$ against the "monomial" functions $(z^n)_{n\in\mathbb{Z}}$ for $z \in S^1$.

In more recent work, the "additive circle" $\mathbb{R}/2\pi\mathbb{Z}$ has been constructed in mathlib as a compact Hausdorff topological group and its Haar measure has been related to the Lebesgue measure on its covering space $\mathbb{R}$, allowing for a refactoring of the construction of Fourier series to treat functions $f : \mathbb{R}/2\pi\mathbb{Z} \to \mathbb{C}$. This unites the computational convenience of the $[0, 2\pi]$ definition with the theoretical convenience of the S^1 definition. This refactor featured work by Oliver Nash [22], Alex Kontorovich and David Loeffler as well as the authors; we will not discuss it further in this article.

As the "monomial" functions $(z^n)_{n\in\mathbb{Z}}$ are central to the definition of Fourier series and the proof of Parseval's identity, we introduce them in Lean in several forms. Firstly we define these functions as a $\mathbb{Z}$-indexed family in the bundled continuous function type $C(S^1, \mathbb{C})$:

```
def fourier (n : ℤ) : C(circle, ℂ) :=
{ to_fun := λ z, z ^ n,
  continuous_to_fun := _ }
```

(we omit the brief proof of their continuity). Later we introduce the unit Haar measure [23] μ on S^1 (in Lean `haar_circle`), from S^1's structure as a topological group, and consider the monomial functions under the name `fourier_Lp p` as elements of the space $L^p(S^1, \mathbb{C})$ (in Lean `Lp ℂ p haar_circle`) with respect to this measure. (A continuous function on a compact topological space equipped with a finite-volume Borel measure is automatically in L^p.)

The linear span of the monomials $(z^n)_{n\in\mathbb{Z}}$ is easily checked to be a subalgebra (i.e., closed under multiplication), to be closed under conjugation, and to separate points. Therefore, by the Stone-Weierstrass theorem, contributed to mathlib by Scott Morrison, the functions `fourier` are dense (in the topology of uniform convergence) in $C(S^1, \mathbb{C})$.

```
lemma span_fourier_closure_eq_top :
  (span ℂ (range fourier)).topological_closure = ⊤
```

By the main theorem of Subsection 9.1, it follows that for each $1 \le p < \infty$, the linear span of the monomials $(z^n)_{n\in\mathbb{Z}}$ is dense (in the L^p topology) in $L^p(S^1, \mathbb{C})$.

```
lemma span_fourier_Lp_closure_eq_top
  {p : ℝ≥0∞} [fact (1 ≤ p)] (hp : p ≠ ∞) :
  (span ℂ (range (fourier_Lp p))).topological_closure = ⊤
```

The case of interest for us is $p = 2$: the space $L^2(S^1, \mathbb{C})$ is a Hilbert space, with the inner product of two functions given by the integral over S^1 of the pointwise quantity $\overline{f(z)}g(z)$. A straightforward computation shows that the monomials $(z^n)_{n\in\mathbb{Z}}$ are an orthonormal set in this Hilbert space.

```
lemma orthonormal_fourier : orthonormal ℂ (fourier_Lp 2)
```

Therefore, by the theory of Subsection 7.2, the family of functions `fourier_Lp 2` can be elevated to a $\mathbb{Z}$-indexed Hilbert basis for $L^2(S^1, \mathbb{C})$:

```
def fourier_series : hilbert_basis ℤ ℂ (Lp ℂ 2 haar_circle)
```

Recall that this is by definition a bijective linear isometry from $L^2(S^1, \mathbb{C})$ to $\ell^2(\mathbb{Z}, \mathbb{C})$, accessed as `fourier_series.repr`. The agreement with the "traditional" definition of the Fourier series of $f \in L^2(S^1, \mathbb{C})$ as a $\mathbb{Z}$-indexed family of complex numbers obtained by integration against $(z^n)_{n\in\mathbb{Z}}$ is a consequence of the lemma `hilbert_basis.repr_apply_apply` from Subsection 7.2:

```
lemma fourier_series_repr (f : Lp ℂ 2 haar_circle) (i : ℤ) :
  fourier_series.repr f i = ∫ z : circle, z ^ (-i) • f z ∂ haar_circle
```

The L^2 convergence of Fourier series is obtained from the lemma `hilbert_basis.has_sum_repr` from Subsection 7.2:

```
lemma has_sum_fourier_series (f : Lp ℂ 2 haar_circle) :
  has_sum (λ i, fourier_series.repr f i • fourier_Lp 2 i) f
```

Finally, Parseval's identity, that the sum of the squared norms of the Fourier coefficients of $f \in L^2(S^1, \mathbb{C})$ equals the L^2 norm of f, is a direct consequence of the fact that `fourier_series.repr` is constructed to be a bijective linear isometry:

```
lemma tsum_sq_fourier_series_repr (f : Lp ℂ 2 haar_circle) :
  ∑' i : ℤ, ‖fourier_series.repr f i‖ ^ 2
  = ∫ z : circle, ‖f z‖ ^ 2 ∂ haar_circle
```

By design, the mathematical arguments in this section are short and translate to Lean without difficulty, requiring less than 200 lines of code in total. The mathematically difficult arguments have all been covered in greater abstraction in the preliminary work, as described in Subsection 7.2 and Subsection 9.1.

10 Frobenius-semilinear maps and isocrystals

Our formal development of semilinear maps was motivated by applications in functional analysis to unify statements and proofs over $\mathbb{R}$ and $\mathbb{C}$. But these maps are interesting and fruitful objects of study in their own right. As an example of an interesting result about semilinear maps that are *not* linear or conjugate-linear, we formalize the one-dimensional case of a theorem of Dieudonné and Manin [9] (see Demazure [24, chapter 4] for a classical exposition and Lurie [25] for a modern outline without proof),

which classifies the isocrystals over an algebraically closed field of characteristic $p > 0$ (Section 2.6).

We denote the ring of `p`-typical Witt vectors over `k` by `𝕎 k` and the field of fractions of this ring by `K(p, k)`. This was defined in `mathlib` by Commelin and Lewis [26], along with the Frobenius endomorphism `frobenius : 𝕎 k →+* 𝕎 k`.

For the remainder of this section, we work in a context where `p` is a prime natural number and `k` is an integral domain of characteristic `p` with a `p`th root function.

```
variables (p : ℕ) [fact p.prime]
  {k : Type*} [comm_ring k] [is_domain k] [char_p k p] [perfect_ring
    k p]
```

Since the base ring `k` has characteristic `p`, `frobenius` satisfies the following property:

```
lemma coeff_frobenius_char_p (x : 𝕎 k) (n : ℕ) :
  (frobenius x).coeff n = (x.coeff n) ^ p
```

The additional hypothesis that `k` has a `p`th root function implies that `frobenius` is in fact an automorphism, and with `k` an integral domain, this induces an automorphism on the field of fractions `K(p, k)`. Locally we let `φ(p, k)` denote this map.

We will be interested in maps between `K(p, k)`-vector spaces that are semilinear in φ ("Frobenius-semilinear"). To facilitate the use of these maps, we add an instance of `ring_hom_inv_pair` (Section 4.3) for φ and its inverse. We also introduce notation `V →ᶠˡ[p, k] V₂` and `V ≃ᶠˡ[p, k] V₂` for the types of Frobenius-semilinear maps and equivalences.

An *isocrystal* is a vector space over the field `K(p, k)` additionally equipped with a Frobenius-semilinear automorphism.

```
class isocrystal (V : Type*) [add_comm_group V] extends module K(p,
    k) V :=
(frob : V ≃ᶠˡ[p, k] V)
```

We denote the map `frob` by `Φ(p, k)`. We say two isocrystals over `K(p, k)` are equivalent (denoted `V ≃ᶠⁱ[p, k] V₂`) if there is a linear equivalence `f : V ≃ₗ[K(p, k)] V₂` which is "Frobenius-equivariant," that is, for all `x`, `Φ(p, k) (f x) = f (Φ(p, k) x)`.

The Dieudonné–Manin theorem classifies the isocrystal structures in every finite dimension, up to this notion of equivalence, over an algebraically closed field `k`. We restrict our attention to the one-dimensional case, where the classification can be stated quite explicitly. The field `K(p, k)` is naturally a vector space over itself with dimension 1. There is a standard family of Frobenius-semilinear automorphisms `K(p, k) ≃ᶠˡ[p, k] K(p, k)` indexed by the integers, namely `p^m · φ(p, k)` for each `m : ℤ`, where the Frobenius automorphism `φ(p, k)` is itself considered as a Frobenius-semilinear automorphism. This induces a $\mathbb{Z}$-indexed family of distinct isocrystals which we refer to as `standard_one_dim_isocrystal p k m`, and we prove that any one-dimensional isocrystal is equivalent to one of the these standard isocrystals.

```
lemma classification [field k] [is_alg_closed k] [char_p k p]
  [add_comm_group V] [isocrystal p k V] (h_dim : finrank K(p, k) V =
    1) :
  ∃ (m : ℤ), nonempty (standard_one_dim_isocrystal p k m ≃ᶠⁱ[p, k] V)
```

The key to proving this statement is finding, for any `a, b : 𝕎 k` with nonzero leading coefficients, a vector `x : 𝕎 k` such that `frobenius x * a = x * b`. We define such an `x` coefficient by coefficient by an intricate recursion that invokes the algebraic closedness of `k` at each step to solve a new polynomial equation. The argument requires us to mediate between different "levels" of polynomials—universal multivariate polynomials over $\mathbb{Z}$, and multivariate and univariate polynomials over `k`—which proved challenging. Arithmetic operations on Witt vectors are notoriously complicated, and the machinery for universal calculations introduced by Commelin and Lewis [26] does not apply here. This key lemma takes 550 lines to establish.

The remainder of the proof of the isocrystal classification theorem was remarkably straightforward. We needed to extend mathlib's Witt vector library to show that when `k` is an integral domain, `𝕎 k` is too. Modulo this and the previous key lemma, the proof (including the definitions of Frobenius-semilinear maps and isocrystals) takes only 100 lines.

The key lemma also constitutes the central argument in establishing that Witt vectors are a *discrete valuation ring*, a theorem we established together with Johan Commelin.

```
instance [field k] [char_p k p] [perfect_ring k p] :
    discrete_valuation_ring (𝕎 k)
```

11 Related work

Given the fundamental importance of linear algebra, it is no surprise that theories have been developed in many proof assistants. To our knowledge, none of these libraries define semilinear maps, none prove the spectral theorem for compact operators, and none prove any of the results we describe generically over $\mathbb{R}$ and $\mathbb{C}$.

Mahmoud, Aravantinos, and Tahar [27] and Afshar et al. [28] both describe developments in HOL Light of complex vector spaces. Both use encodings inherently specific to the complex case; they do not generalize the work over the reals by Harrison [29].

Aransay and Divasón [30] introduce vector spaces over arbitrary fields to Isabelle/HOL, using a careful combination of type classes and Isabelle's *locale* feature. A paper by the same authors [10] describes an experiment to generalize the Isabelle definition of a real inner product space to a larger class of fields, using a type class that seems analogous to our class `is_R_or_C` (Section 5.1). Implementing this idea systematically would probably involve providing a locale-based generalization of *euclidean-space* at the beginning of the Isabelle/HOL mathematical analysis library, and the authors do not take this project on, despite noting how useful the generalization would be.

An Isabelle Archive of Formal Proofs entry by Caballero and Unruh [19] duplicates much of the real vector space development in the complex setting, in the process introducing conjugate-linear maps and the complex adjoint operator. Little infrastructure seems to be shared between the real and complex cases. Their development includes a proof of Fréchet–Riesz over $\mathbb{C}$, but does not indicate how it might specialize to $\mathbb{R}$. Also motivated by applications in quantum computation, Bordg et al. [31] define the conjugate-transpose, the analogue of the adjoint in the matrix setting, but again do not generalize to arbitrary fields.

Perhaps related to the more expressive type theory, Coq developments of linear algebra have taken more advantage of type polymorphism. The Mathematical Components library [32] features a theory of modules over arbitrary scalar rings, as does Coquelicot [5]. Building on both these libraries, MathComp-Analysis [33] develops structures used in functional analysis. A `linear_for` predicate in Mathematical Components expresses a concept which is mathematically slightly more general than our `semilinear_map` definition, but which has less convenient properties under composition and inversion. In a branch of the Mathematical Components repository,[6] Cohen defines Hermitian forms, which diverge in behavior over $\mathbb{R}$ and $\mathbb{C}$ similar to conjugate-linear maps. The approach here has some similarities to ours, but preserves fewer definitional equalities; in particular, our conjugate-linear maps on $\mathbb{R}$ are definitionally linear maps, while the analogous statement does not hold for the Mathematical Components approach to Hermitian forms.

Boldo et al. [17] prove the real case of Fréchet–Riesz using Coquelicot, on the way to the Lax–Milgram theorem, but do not address the complex case. Narita et al. [18] do the same in Mizar. Cohen proves the diagonalization theorem for normal matrices in the same of the Mathematical Components repository.[6] This is mathematically equivalent to the diagonalization theorem for normal endomorphisms of a finite-dimensional space described at the start of Section 8.2. Cohen's matrix version could more easily be converted for use in verified numerical analysis, whereas the abstract linear-map version we provide is more convenient in mathematical applications and also admits a more streamlined proof.

An extensive development of Fourier series was written by Harrison in HOL Light[7], and has been ported to Isabelle [34] by Paulson. This treatment covers quite a bit of the same ground as ours, notably the theorem on L^2 convergence of Fourier series. It also covers numerous topics not treated by us, including the Riemann-Lebesgue lemma (this is also ongoing work in Lean of David Loeffler), Dini's test and Cesaro summability. It is a much more concrete treatment, proving numerous results in the special setting of $L^2([0, 2\pi], \mathbb{R})$ which we deduce from the abstract theory of Hilbert bases. Notably, it appears not to contain Parseval's identity, which comes effectively for free from our development of Hilbert space theory. A further minor difference is that this development takes place over $\mathbb{R}$ (with trigonometric functions) rather than over $\mathbb{C}$ (with exponential/monomial functions), due to the restriction to real scalars in these languages' functional analysis libraries.

A Metamath treatment of Fourier series by Glauco Siliprandi[8] has less overlap with ours, primarily treating Fourier series of continuous functions.

Supplementary information. We maintain a guide through the code corresponding to this paper at https://robertylewis.com/semilinear-paper/.

Acknowledgments. We thank Johan Commelin for many conversations about isocrystals and Johannes Hölzl for comments on work in Isabelle. We thank the mathlib community and maintainer team for insightful comments and suggestions during code review.

[6] https://github.com/math-comp/math-comp/pull/207
[7] https://github.com/jrh13/hol-light/blob/master/100/fourier.ml
[8] https://us.metamath.org/mpeuni/fourier.html

References

[1] Lammich, P., Lochbihler, A.: Automatic refinement to efficient data structures: A comparison of two approaches. J. Autom. Reason. **63**(1), 53–94 (2019) https://doi.org/10.1007/s10817-018-9461-9

[2] Wirth, N.: Program development by stepwise refinement. Commun. ACM **14**(4), 221–227 (1971) https://doi.org/10.1145/362575.362577

[3] Hölzl, J., Immler, F., Huffman, B.: Type classes and filters for mathematical analysis in Isabelle/HOL. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7998, pp. 279–294. Springer, ??? (2013). https://doi.org/10.1007/978-3-642-39634-2_21 . https://doi.org/10.1007/978-3-642-39634-2_21

[4] Affeldt, R., Cohen, C., Rouhling, D.: Formalization techniques for asymptotic reasoning in classical analysis. J. Formalized Reasoning **11**(1), 43–76 (2018) https://doi.org/10.6092/issn.1972-5787/8124

[5] Boldo, S., Lelay, C., Melquiond, G.: Coquelicot: A user-friendly library of real analysis for Coq. Mathematics in Computer Science **9**(1), 41–62 (2015) https://doi.org/10.1007/s11786-014-0181-1

[6] Buzzard, K., Commelin, J., Massot, P.: Formalising perfectoid spaces. In: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs. CPP 2020, pp. 299–312. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372885.3373830

[7] The mathlib Community: The Lean mathematical library. In: CPP, pp. 367–381. ACM, New York, NY, USA (2020). https://doi.org/10.1145/3372885.3373824

[8] Moura, L., Kong, S., Avigad, J., Doorn, F., Raumer, J.: The Lean Theorem Prover (system description). In: Felty, A.P., Middeldorp, A. (eds.) CADE-25, pp. 378–388. Springer, Cham (2015)

[9] Manin, J.I.: Theory of commutative formal groups over fields of finite characteristic. Uspehi Mat. Nauk **18**(6 (114)), 3–90 (1963)

[10] Aransay, J., Divasón, J.: A formalisation in HOL of the fundamental theorem of linear algebra and its application to the solution of the least squares problem. J. Autom. Reason. **58**(4), 509–535 (2017) https://doi.org/10.1007/s10817-016-9379-z

[11] Dupuis, F., Lewis, R.Y., Macbeth, H.: Formalized functional analysis with semilinear maps. In: Andronick, J., Moura, L. (eds.) 13th International Conference on Interactive Theorem Proving (ITP 2022). Leibniz International Proceedings in Informatics (LIPIcs), vol. 237, pp. 10–11019. Schloss Dagstuhl – Leibniz-Zentrum

für Informatik, Dagstuhl, Germany (2022). https://doi.org/10.4230/LIPIcs.ITP.2022.10 . https://drops.dagstuhl.de/opus/volltexte/2022/16719

[12] Hazewinkel, M.: Witt vectors. Part 1. Handbook of Algebra, 319–472 (2009) https://doi.org/10.1016/s1570-7954(08)00207-6

[13] Spitters, B., Weegen, E.: Type classes for mathematics in type theory. Mathematical Structures in Computer Science **21**(4), 795–825 (2011) https://doi.org/10.1017/S0960129511000119

[14] Baanen, A.: Use and abuse of instance parameters in the Lean mathematical library. In: Andronick, J., Moura, L. (eds.) 13th International Conference on Interactive Theorem Proving (ITP 2022). Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2022)

[15] Doorn, F., Ebner, G., Lewis, R.Y.: Maintaining a library of formal mathematics. In: Benzmüller, C., Miller, B. (eds.) Intelligent Computer Mathematics, pp. 251–267. Springer, Cham (2020)

[16] Wieser, E.: Scalar actions in Lean's mathlib. CoRR **abs/2108.10700** (2021) 2108.10700

[17] Boldo, S., Clément, F., Faissole, F., Martin, V., Mayero, M.: A Coq formal proof of the Lax–Milgram theorem. In: Bertot, Y., Vafeiadis, V. (eds.) Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, January 16-17, 2017, pp. 79–89. ACM, ??? (2017). https://doi.org/10.1145/3018610.3018625 . https://doi.org/10.1145/3018610.3018625

[18] Narita, K., Endou, N., Shidama, Y.: The orthogonal projection and the Riesz representation theorem. Formaliz. Math. **23**(3), 243–252 (2015) https://doi.org/10.1515/forma-2015-0020

[19] Caballero, J.M.R., Unruh, D.: Complex bounded operators. Archive of Formal Proofs (2021). https://isa-afp.org/entries/Complex_Bounded_Operators.html, Formal proof development

[20] Gouëzel, S.: Lp spaces. Archive of Formal Proofs (2016). https://isa-afp.org/entries/Lp.html, Formal proof development

[21] Kudryashov, Y.: Formalizing the divergence theorem and the Cauchy integral formula in Lean. In: Andronick, J., Moura, L. (eds.) 13th International Conference on Interactive Theorem Proving (ITP 2022). Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2022)

[22] Nash, O.: A Formalisation of Gallagher's Ergodic Theorem (2023)

[23] Doorn, F.: Formalized Haar Measure. In: Cohen, L., Kaliszyk, C. (eds.) 12th International Conference on Interactive Theorem Proving (ITP 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 193, pp. 18–11817. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.ITP.2021.18 . https://drops.dagstuhl.de/opus/volltexte/2021/13913

[24] Demazure, M.: Lectures on p-Divisible Groups. Lecture Notes in Mathematics. Springer, ??? (2006)

[25] Lurie, J.: Lecture Notes on the Fargues–Fontaine Curve. Lecture 26: Isocrystals (2018). https://www.math.ias.edu/\texttildelowlurie/205notes/Lecture26-Isocrystals.pdf

[26] Commelin, J., Lewis, R.Y.: Formalizing the ring of Witt vectors. In: Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs. CPP 2021, pp. 264–277. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3437992.3439919 . https://doi.org/10.1145/3437992.3439919

[27] Mahmoud, M.Y., Aravantinos, V., Tahar, S.: Formalization of infinite dimension linear spaces with application to quantum theory. In: Brat, G., Rungta, N., Venet, A. (eds.) NASA Formal Methods, 5th International Symposium, NFM 2013, Moffett Field, CA, USA, May 14-16, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7871, pp. 413–427. Springer, ??? (2013). https://doi.org/10.1007/978-3-642-38088-4_28 . https://doi.org/10.1007/978-3-642-38088-4_28

[28] Afshar, S.K., Aravantinos, V., Hasan, O., Tahar, S.: Formalization of complex vectors in higher-order logic. In: Watt, S.M., Davenport, J.H., Sexton, A.P., Sojka, P., Urban, J. (eds.) Intelligent Computer Mathematics - International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8543, pp. 123–137. Springer, ??? (2014). https://doi.org/10.1007/978-3-319-08434-3_10 . https://doi.org/10.1007/978-3-319-08434-3_10

[29] Harrison, J.: The HOL Light theory of Euclidean space. J. Autom. Reason. **50**(2), 173–190 (2013) https://doi.org/10.1007/s10817-012-9250-9

[30] Aransay, J., Divasón, J.: Generalizing a mathematical analysis library in Isabelle/HOL. In: Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9058, pp. 415–421. Springer, ??? (2015). https://doi.org/10.1007/978-3-319-17524-9_30 . https://doi.org/10.1007/978-3-319-17524-9_30

[31] Bordg, A., Lachnitt, H., He, Y.: Certified quantum computation in Isabelle/HOL. J. Autom. Reason. **65**(5), 691–709 (2021) https://doi.org/10.1007/s10817-020-09584-7

[32] Mahboubi, A., Tassi, E.: Mathematical Components. Zenodo, ??? (2020). https://doi.org/10.5281/zenodo.4282710

[33] Affeldt, R., Cohen, C., Kerjean, M., Mahboubi, A., Rouhling, D., Sakaguchi, K.: Competing inheritance paths in dependent type theory: A case study in functional analysis. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12167, pp. 3–20. Springer, ??? (2020). https://doi.org/10.1007/978-3-030-51054-1_1 . https://doi.org/10.1007/978-3-030-51054-1_1

[34] Paulson, L.C.: Fourier series. Archive of Formal Proofs (2019). https://isa-afp.org/entries/Fourier.html, Formal proof development