A Comprehensive Overview of the Lebesgue Differentiation Theorem in Coq

Reynald Affeldt
National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

Zachary Stone
The MathComp-Analysis development team, Boston, MA, USA

Abstract

Formalization of real analysis offers a chance to rebuild traditional proofs of important theorems as unambiguous theories that can be interactively explored. This paper provides a comprehensive overview of the Lebesgue Differentiation Theorem formalized in the Coq proof assistant, from which the first Fundamental Theorem of Calculus (FTC) for the Lebesgue integral is obtained as a corollary. Proving the first FTC in this way has the advantage of decomposing into loosely-coupled theories of moderate size and of independent interest that lend themselves well to incremental and collaborative development. We explain how we formalize all the topological constructs and all the standard lemmas needed to eventually relate the definitions of derivability and of Lebesgue integration of MathComp-Analysis, a formalization of analysis developed on top of the Mathematical Components library. In the course of this experiment, we substantially enrich MathComp-Analysis and even devise a new proof for Urysohn's lemma.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Type theory; Mathematics of computing → Mathematical analysis

Keywords and phrases Coq proof assistant, Mathematical Components library, Lebesgue integral, fundamental theorem of calculus

Digital Object Identifier 10.4230/LIPIcs.ITP.2024.5

Supplementary Material *Software (Source Code)*: `https://github.com/math-comp/analysis` [1]
archived at `swh:1:dir:603335be2da03f1a37b210f027047f94aa0ed2c6`

Funding *Reynald Affeldt*: The first author acknowledges support of the JSPS KAKENHI Grant Number 22H00520.

Acknowledgements The authors are grateful to A. Bruni, C. Cohen, Y. Ishiguro, and T. Saikawa for their inputs. This work has benefited from feedback gained during the MathComp-Analysis development meetings. The authors would like to thank the anonymous referees of ITP 2024 for their careful and informative review, as well as the program committee of JFLA 2024 where a preliminary version of this work was presented [5].

1 Introduction

The formalization of the Fundamental Theorem of Calculus (FTC) for the Lebesgue integral in the Coq proof assistant [29] is an ongoing work as part of the development of MathComp-Analysis [1], a library for analysis that extends the Mathematical Components library [17] with classical axioms [3, §5]. Besides mathematics, MathComp-Analysis has also been used to formalize programming languages [4, 27, 31].

The first FTC for Lebesgue integration can be stated as follows: for f integrable on $\mathbb{R}$, $F(x) \overset{\text{def}}{=} \int_{t\in]-\infty,x]} f(t)(\mathbf{d}\,\mu)$ is differentiable and $F'(x) = f(x)$ almost-everywhere (hereafter, a.e.) relatively to μ, where μ is the Lebesgue measure. This is different from the standard statement for the Riemann integral, where f is assumed to be continuous, making for a simple proof. In comparison, connecting derivation and Lebesgue integration under an integrability hypothesis is unwieldy, even more so in MathComp-Analysis whose formalizations of derivation [3, §4.5] and of the Lebesgue integral [2, §6.4] have been unrelated so far. They can be bridged thanks to the Lebesgue Differentiation theorem and this is appealing for two reasons. First, it is a useful theorem in itself: the first FTC is a consequence, as well as other results such as Lebesgue's density theorem. Second, we can decompose its proof in several results: this provides a way to incrementally enrich the theories of MathComp-Analysis. We think that this approach is an instance of a more generic way to tackle formalization of mathematics: find a path through the literature to present many key results as easy consequences of a central, technical lemma with rather weak assumptions. Incidentally, such a fine-grained decomposition also provides a practical way to monitor formalization progress.

In terms of formalization in a proof assistant, our contributions are as follows:

- We provide the first formalization of the first FTC for Lebesgue integration in Coq.
- We bring to Coq several standard lemmas and theorems of measure theory: Vitali's lemmas and theorem, a theory of Hardy-Littlewood's operator, Urysohn's lemma, Ergorov's, Lusin's, Tietze's theorems, and the Lebesgue Differentiation theorem. In particular, among these results, Urysohn's lemma is given an original proof. We also improve the MathComp-Analysis support for topology (lower semicontinuity, normal spaces, subspaces, etc.) and for real functions (by extending the theory for $\limsup$ / $\liminf$). The formalization of the first FTC for Lebesgue integration provides a strong evidence that these pieces of formalization can indeed be combined to achieve a large result.

Another intent with this paper is to produce an informative document for potential users of the topology and measure theories of MathComp-Analysis. The whole library is still under development in the sense that not all notions are formalized as we would like them to be, often to cope with temporary limitations of the available tooling. Yet, we did observe that it is already a useful tool. For example, we could use it to revisit the proof of Urysohn's lemma by producing an original proof. Also, we had to clarify a few proof steps that are often hand-waved in lecture notes: typically, generalizations from lemmas stated for bounded cases only. Filling such gaps is almost business-as-usual when formalizing mathematics, but we believe that it is important to document them to better anticipate similar gaps in the future. For these reasons, we think that our formalization of the FTC provides a nice milestone to document MathComp-Analysis.

Outline

Regarding mathematical proofs, we stay at the level of a bird's-eye view and instead focus on the main aspects of the formalization. We start by recalling the basics of MathComp-Analysis in Sect. 2. We explain the formal statement of the Lebesgue Differentiation theorem in Sect. 3 and provide an overview of its proof in Sect. 4. To formalize this proof, we extend MathComp-Analysis with new topological constructs in Sect. 5 and with basic but new measure-theoretic lemmas in Sect. 6. In the particular case of Urysohn's lemma, we explain the formalization of an original proof in Sect. 7. The main intermediate lemmas of the Lebesgue Differentiation theorem are the purpose of Sect. 8 and Sect. 9. Finally, we apply the Lebesgue Differentiation theorem to the proof of the first FTC for Lebesgue integration and to the proof of Lebesgue's density theorem in Sect. 10. We review related work in Sect. 11 and conclude in Sect. 12.

2 Background on MathComp-Analysis

MathComp-Analysis is built on top of MathComp and reuses several of its theories. We use in particular the following notations, which are traditionally in ASCII in MathComp libraries. The successor function of natural numbers is noted `.+1`, multiplication of a natural number by 2 is noted `.*2`. A multiple conjunction is noted `[/\ P1, P2, ... & Pn]`. Function composition is noted `\o`. Point-wise equality between two functions is noted `=1`. Point-wise multiplication of two functions `f` and `g` is noted `f \* g`. The notation `f ^~ y` is for the function $\lambda x.\mathtt{f}\,x\,\mathtt{y}$. Intervals are noted `` `]a, b[ ``, `` `]a, b] ``, etc. One can inject a natural number `n` into a ring with the notation `n%:R`. The inverse of a field element `r` is noted `r^-1`. The norm of `x` is noted `` `|x| ``. The inclusion between two lists `s` and `r` is noted `{subset s <= r}`. We write `x \in s` when an element `x` belongs to the list `s`.

MathComp-Analysis comes with library support for set theory. Given a type `T`, `set T` is the type of sets of elements of type `T`. The notation `[set: T]` is for the set of all the elements of type `T`; the singleton set containing `x` is noted `[set x]`; and `set0` represents the empty set. To unambiguously improve readability, we use standard LaTeX notations instead of ASCII for the set-theoretic operations set inclusion ($\subseteq$), set intersection ($\cap$), set union ($\cup$), set difference ($\setminus$), and the product of sets ($\times$). The complement of a set `A` is noted $\mathtt{A}^{\mathtt{C}}$. (If necessary, see [2, Table 2] for a list-up of the corresponding ASCII notations.) Set difference with a singleton is noted `` A `\ x ``: it is a shortcut for `A` $\setminus$ `[set x]`. The image by a function `f` of a set `A` is written `` f @` A ``. The set $\{f(x) \mid x \in A\}$ defined by comprehension is noted `[set f x | x in A]`. A list `s` can be turned into a set with the notation `` [set` s] ``. The notation `A !=set0` means that the set `A` is not empty. A family `F` of pairwise-disjoint sets indexed by `D` is noted `trivIset D F`. The characteristic function over a set `A` is noted `\1_A`.

MathComp-Analysis extends the numeric types of MathComp with the type `realType` for reals. When `R` is a numeric type, `\bar R` is the numeric type extended with `-oo` and `+oo`, so that when `R : realType`, `\bar R` corresponds to $\overline{\mathbb{R}}$. One can inject a numeric value `r` into the corresponding type of extended numbers with the notation `r%:E` (this is actually a notation for `EFin r`). An extended number `x` can be projected to the corresponding numeric value by `fine x` (which is `0` when `x` is $\pm\infty$). The supremum of a set of extended reals `A` is noted `ereal_sup A`. In this paper, the variable `R` has type `realType` unless stated otherwise.

Like several other libraries [7, 13, 30], MathComp-Analysis uses filters to formalize topology. For example, we note `\oo` the filter consisting of the set of predicates over natural numbers that are eventually true; `x^'` the *deleted neighborhood filter* of `x`, i.e., the set of neighborhoods of `x` from which `x` is excluded; `x^'+` for *right filters*, i.e., the filters of neighborhoods of `x` intersected with $]\mathtt{x}, +\infty[$, and similarly for the *left filters* note `x^'-`. Filters are associated to elements of a given type upon the definition of a *filtered type*. The convergence statement $f(x) \xrightarrow[x\to a]{} \ell$ is noted `f x @[x --> a] --> l`; the limit of a filter `F` is noted `lim F` [3, §2.3]. Topological spaces are built on top of filtered types and their type is `topologicalType`. In a topological space, the set of neighborhoods of `x` is `nbhs x`. Mathematical structures such as topological spaces are defined and instantiated using a Coq extension called Hierarchy-Builder [9]. With this tool, interfaces are defined as so-called *factories* whose definition generates constructors to build instances. See [2, §3.1] for a quick reference to Hierarchy-Builder. The predicates `open`, `closed`, `closure`, and `compact` correspond to the eponymous topological notions. The type of uniform spaces is `uniformType`. Given a uniform space `M`, entourages are objects of type `set (set (M * M))` that satisfy the axioms of uniform space for `M`. One of the axiom of uniform spaces guarantees that for each entourage `E`, there is an entourage `V` with $\{(x, z) \mid \exists y,\ (x, y) \in V \wedge (y, z) \in V\} \subseteq E$. We denote this entourage with `split_ent(E)`. MathComp-Analysis also provides pseudometric spaces in which a ball is noted `ball`; over the real line, a ball is a centered open interval. The type of sequences (indexed by natural numbers) over `T` is noted `T^nat`.

The basics of measure theory in MathComp-Analysis is documented in previous work [2]. A `measurableType` is a type equipped with a structure of σ-algebra. Given a measurable type `T` and `A` of type `set T`, we write `measurable A` when the set `A` is measurable. The fact that the extended real-valued function f is measurable over D is noted `measurable_fun D f`. For a measure `mu`, the fact that `f` is integrable over `D` is noted `mu.-integrable D f`. The integral $\int_{x \in A} f(x)(\mathbf{d}\,\mu)$ is noted `\int[mu]_(x in A) f x`. When `A` is negligible for a measure `mu`, we write `mu.-negligible A`. The fact that the predicate `P` holds a.e. relatively to `mu` is noted `{ae mu, forall x, P x}`.

3 Statement of the Lebesgue Differentiation theorem

Let us note $[f]_A \overset{\text{def}}{=} \frac{1}{\mu(A)} \int_{y \in A} |f(y)|(\mathbf{d}\,\mu)$ the average of a real-valued function f over the set A. Using the existing notations of MathComp-Analysis, we formalize this notion as follows:

```
Definition iavg (f : R -> R) (A : set R) :=
  (fine (mu A))^-1%:E * \int[mu]_(y in A) `| (f y)%:E |.
```

Recall from Sect. 2 that `%:E` injects a numeric value into its extended version and that `fine` performs a corresponding projection. We also introduce the notation $\overline{f_{\mathrm{B}_r(x)}} \overset{\text{def}}{=} [\lambda y. f(y) - f(x)]_{\mathrm{B}_r(x)}$ where $\mathrm{B}_r(x)$ is a ball centered at x of radius r:

```
Definition davg (f : R -> R) (x r : R) := iavg (center (f x) \o f) (ball x r).
```

The `center c` function is $\lambda y. y - c$.

Given a real-valued function f, a *Lebesgue point* is a real number x s.t. $\overline{f_{\mathrm{B}_r(x)}} \xrightarrow[r \to 0^+]{} 0$. Reusing the Lebesgue measure (hereafter `mu`) formalized in previous work [2, §5.2], we formally define Lebesgue points:

```
Definition lebesgue_pt (f : R -> R) (x : R) :=
  davg f x r @[r --> 0^'+] --> 0.
```

Note the use of right filters (Sect. 2) to define the fact that `r` tends to 0^+. The Lebesgue Differentiation theorem states that, for a real-valued, locally-integrable function f (i.e., integrable on compact subsets of its domain), we have Lebesgue points a.e.:

```
Lemma lebesgue_differentiation (f : R -> R) : locally_integrable [set: R] f
  -> {ae mu, forall x, lebesgue_pt f x}.
```

Being locally integrable can be defined as the following conjunction:

```
Definition locally_integrable (D : set R) (f : R -> R) :=
  [/\ measurable_fun D f, open D & forall K, K ⊆ D -> compact K ->
        \int[lebesgue_measure]_(x in K) `|f x|%:E < +oo].
```

In fact, this definition specializes in a locally compact space such as $\mathbb{R}$ to the conjunction of `open D` and

```
forall x, D x -> exists U, open_nbhs x U /\ mu.-integrable U (EFin \o f)
```

where `open_nbhs` is a predicate for open neighborhoods; yet, we stuck to the previous definition from our reference textbook [16].

4 Proof of the Lebesgue Differentiation theorem

The first step of the proof of the Lebesgue Differentiation theorem is to reduce the problem to functions $f_k \stackrel{\text{def}}{=} f\mathbb{1}_{B_k}$ with $B_k \stackrel{\text{def}}{=} \mathrm{B}_{2(k+1)}(0)$:

```
Lemma lebesgue_differentiation_bounded (f : R -> R) :
  let B k := ball 0 k.+1.*2%:R in let f_ k := f \* \1_(B k) in
  (forall k, mu.-integrable [set: R] (EFin \o f_ k)) ->
  forall k, {ae mu, forall x, B k x -> lebesgue_pt (f_ k) x}.
```

This problem reduction is often hand-waved in lecture notes (Schwartz's presentation is one exception [28, eqn (5.12.101)]).

Second, instead of proving for all k that we have a.e. Lebesgue points over B_k, it is sufficient to prove that the set $A_k(a) \stackrel{\text{def}}{=} B_k \cap \left\{x \,|\, a < \limsup_{r\to 0} \overline{f_k}_{\mathrm{B}_r(x)}\right\}$ is negligible for all $a > 0$, i.e.:

```
(* local context omitted *)
==============================
mu.-negligible (B k ∩ [set x | a%:E < (f_ k)^* x])
```

where h`^* x` is a (local) notation for $\limsup_{r\to 0} \overline{h_{\mathrm{B}_r(\mathtt{x})}}$.

For this last step, the idea is to exhibit a sequence of continuous functions g_i such that

$$A_k(a) \subseteq \bigcap_n B_k \cap \Big(\underbrace{\{x \,|\, f_k(x) - g_n(x) \geq a/2\}}_{(a)} \cup \underbrace{\{x \,|\, \mathrm{HL}(f_k(x) - g_n(x)) > a/2\}}_{(b)}\Big)$$

where $\mathrm{HL}(f)(x) \stackrel{\text{def}}{=} \sup_{r>0}\{[f]_{\mathrm{B}_r(x)}\}$ is the Hardy-Littlewood operator.We can show that the measure of the right-hand side is null. To deal with (a), we use Markov's inequality (i.e., the fact that $\mu(\{x \,|\, |f(x)| \geq a\}) \leq \frac{1}{a}||f||_1$ for $a > 0$ and a measure μ, where $||\cdot||_1$ is the L^1 norm) and the fact that continuous functions are dense in L^1, whose proof requires several standard results of measure theory (in particular, Urysohn's lemma). To deal with (b), we need the Hardy-Littlewood maximal inequality, which in turns relies on Vitali's covering lemma.

Figure 1 shows the main lemmas that we add to MathComp-Analysis to formalize this proof.

5 Topological constructs added to MathComp-Analysis

Before explaining the formalization of the main lemmas to prove the Lebesgue Differentiation theorem, we explain the preparatory work needed to extend the formalization of topology of MathComp-Analysis. In particular, as a preliminary step to be able to state Egorov's theorem, Lusin's theorem, and Tietze's theorem, we extend MathComp-Analysis with the subspace topology and with the topology of uniform convergence.

5.1 Subspace topologies

Given a type T equipped with a topology $\mathcal{T}$ and a set A of elements of T, $\{A \cap U \mid U \in \mathcal{T}\}$ forms a subspace topology. We formalize subspace topologies on the basis of the existing formalization of topological space of MathComp-Analysis and using Hierarchy-Builder (see Sect. 2). First, we introduce an identifier `subspace` that serves as an alias for a type `T` and which is parameterized by a set `A`:

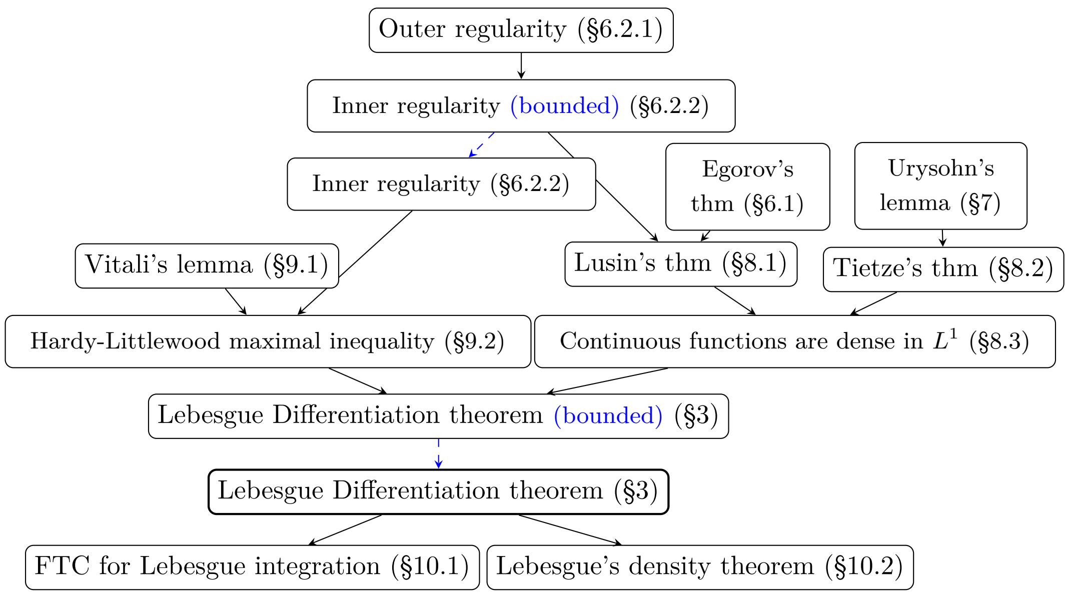
Outer regularity (§6.2.1)
Inner regularity (bounded) (§6.2.2)
Inner regularity (§6.2.2)
Egorov's thm (§6.1)
Urysohn's lemma (§7)
Vitali's lemma (§9.1)
Lusin's thm (§8.1)
Tietze's thm (§8.2)
Hardy-Littlewood maximal inequality (§9.2)
Continuous functions are dense in L^1 (§8.3)
Lebesgue Differentiation theorem (bounded) (§3)
Lebesgue Differentiation theorem (§3)
FTC for Lebesgue integration (§10.1)
Lebesgue's density theorem (§10.2)

Figure 1 Overview of the proof of the Lebesgue Differentiation theorem and derived results. (dashed arrows indicate generalizations from statements about bounded sets)

```
Definition subspace {T : Type} (A : set T) := T.
```

When the underlying type `T` is equipped with a topology (i.e., it has type `topologicalType`), we can equip the identifier `subspace` with a structure of topology relative to `A`. For that purpose, we start by defining a filter for a point `x` in the subspace topology as: (1) the restriction of the neighborhoods of `x` to `A` (below: `within A (nbhs x)`) when `x` belongs to `A`, or, otherwise, (2) the filter of the sets containing the singleton $\{x\}$ (below: `globally [set x]`):

```
Definition nbhs_subspace (x : subspace A) : set_system (subspace A) :=
  if x \in A then within A (nbhs x) else globally [set x].
```

A `set_system` is simply synonymous for a set of sets. Filters defined in this way give rise to a filtered type (Sect. 2). To attach the structure of filtered type with the identifier `subspace`, it suffices to summon Hierarchy-Builder with the appropriate *factory*:

```
HB.factory Record hasNbhs T := { nbhs : T -> set_system T }.
```

A factory is represented by an interface (here, `hasNbhs`) and its definition gives rise to a constructor (here, `hasNbhs.Build`) that can be used to build instances, e.g.:

```
HB.instance Definition _ := hasNbhs.Build (subspace A) nbhs_subspace.
```

The filtered type associated with `subspace` can furthermore be equipped with a topology using this other factory provided by MathComp-Analysis:

```
HB.factory Record Nbhs_isNbhsTopological T of Nbhs T := {
  nbhs_filter : forall p : T, ProperFilter (nbhs p);
  nbhs_singleton : forall (p : T) (A : set T), nbhs p A -> A p;
  nbhs_nbhs : forall (p : T) (A : set T), nbhs p A -> nbhs p (nbhs ^~ A) }.
```

Indeed, (1) one can show that `nbhs_subspace`'s are *proper filters* (i.e., no set in the filter is empty) [3, §3.2.2], (2) points are contained into their neighborhoods, and (3) given a neighborhood A, the set of points of which A is a neighborhood is also a neighborhood. It suffices to build an instance of topological type by passing to the factory above the proofs of the facts (1), (2), and (3) (omitted here but that can be found in the formal development [1, file `topology.v`]):

```
HB.instance Definition _ := Nbhs_isNbhsTopological.Build (subspace A)
  nbhs_subspace_filter nbhs_subspace_singleton nbhs_subspace_nbhs.
```

This is the subspace topology relative to `A`. Note that this topology is discrete outside of `A`.

In particular, we use the topology of subspace to define continuity as follows. The notation `{within A, continuous f}` denotes continuity of `f : subspace A -> Y`:

```
Notation "{ 'within' A , 'continuous' f }" :=
  (continuous (f : subspace A -> _)).
```

The notation `continuous` comes from MathComp-Analysis [1, file `topology.v`] and predates this work. The purpose of the notation `{within A, continuous f}` is to ensure that the continuity of `f` depends only on its values in `A` while `f(x + eps)` type-checks. If `x` and `eps` have type `subspace A`, then `f(x + eps)` is indeed well-defined. One might naively attempt to use the sigma type `{x | A x}` with the weak topology (i.e., the topology defined by the preimages of opens for a given function) induced by the function `projT1 : {x : T | A x} -> T` to define the subspace topology. However, there is no clear way to define addition for `{x | A x}` so that `f(x + eps)` is well-typed.

5.2 Uniform convergence

The methodology to formalize the topology of uniform convergence is similar to the one used to formalize subspaces.

First, let us recall how total functions are equipped with the structure of uniform space in MathComp-Analysis. There is an identifier `arrow_uniform_type` which is an alias for the type `U -> V` with `U : choiceType` and `V : uniformType`. The corresponding uniform space is generated by the entourages $\{(f, g) \mid \forall x : U, E(f(x), g(x))\}$ where E is an entourage of V (see `fct_ent` in [1, file `function_spaces.v`]). As a consequence of the appropriate Hierarchy-Builder instantiation, `arrow_uniform_type` is given the type `uniformType`.

Now, we formalize a topology whose elements are functions from the set `A` to the type `V`. First, we introduce an identifier:

```
Definition uniform_fun {U : Type} (A : set U) (V : Type) : Type := U -> V.
```

We also introduce a notation ``{uniform` A -> V}`` which stands for `@uniform_fun _ A V`. We then introduce the (high-order) function `sigL_arrow` that turns a function $U \to V$ between two types into a function $A \to V$ from a set to a type:

```
Definition sigL_arrow {U : choiceType} (A : set U) (V : uniformType) :
  (U -> V) -> arrow_uniform_type A V := @sigL _ V A.
```

The function `sigL` comes from [1, file `functions.v`]. The identifier `uniform_fun` is then equipped with the weak topology (`weak_topology` in MathComp-Analysis) induced by `sigL_arrow` and eventually the desired topology is simply obtained by *copying* the structure obtained by weak topology:

```
HB.instance Definition _ (U : choiceType) (A : set U) (V : uniformType) :=
  Uniform.copy {uniform` A -> V} (weak_topology (@sigL_arrow _ A V)).
```

Copying a structure is a feature provided by HIERARCHY-BUILDER. Note that we can copy the structure using `Uniform.copy` instead of `Topological.copy` because weak topology always inherits a uniform structure, see the section `weak_uniform` in [1, file `topology.v`].

Then, the uniform convergence of a sequence of functions `F` towards `f` over a set `A` can be defined. It is the filter inclusion of the neighborhoods of `f` in `{uniform` A -> V}` into the filter `F`, and this inclusion is written `cvg_to F (nbhs (f : {uniform` A -> _}))` in MATHCOMP-ANALYSIS. The notation `{uniform A, F --> f}` is for the latter inclusion.

5.3 More support for extended real-valued functions

Besides the topological structures we explained in the previous sections, Sect. 4 also highlights the need to generalize MATHCOMP-ANALYSIS's theory of lim sup / lim inf. Before our experiment, this theory was limited to sequences (indexed by natural numbers) that were actually introduced to develop the monotone convergence theorem and its consequences [2, §6.5]. Handling extended real-valued functions over the real numbers requires the formalization of the following definition: $\limsup_{x \to a} f(x) \overset{\text{def}}{=} \lim_{\varepsilon \to 0^+} \sup\{f(x) \,|\, x \in \mathrm{B}_\varepsilon(a) \setminus \{a\}\}$. It can be couched in formal terms by first defining the limit superior of a function `f` at filter `F`:

```
Variables (T : choiceType) (X : filteredType T) (R : realFieldType).
Implicit Types (f : X -> \bar R) (F : set (set X)).
Definition limf_esup f F := ereal_inf [set ereal_sup (f @` V) | V in F].
```

We can then specialize this definition to define the limit superior of a function over the type of real numbers by using deleted neighborhood filters:

```
Variable R : realType.
Implicit Types (f : R -> \bar R) (a : R).
Definition lime_sup f a : \bar R := limf_esup f a^'.
```

This generic definition of `lime_sup` can be shown to be equivalent to $\lim_{\varepsilon \to 0^+} \sup\{f(x) \,|\, x \in \mathrm{B}_\varepsilon(a) \setminus \{a\}\}$:

```
Let sup_ball f a r := ereal_sup [set f x | x in ball a r `\ a].
Lemma lime_sup_lim f a : lime_sup f a = lim (sup_ball f a e @[e --> 0^'+]).
```

In the course of formalizing the Lebesgue Differentiation theorem, developing the theory of lim sup/lim inf turned out to be a non-trivial intermission, which revealed some quirks (now fixed) in the automatic handling of right filters using the `near` tactics [3, §3.2] in MATHCOMP-ANALYSIS.

6 Egorov's theorem and regularity

The top part of Fig. 1 reveals the first measure-theoretic results needed to prove the Lebesgue Differentiation theorem. Lusin's theorem requires Egorov's theorem as well as the inner regularity of the Lebesgue measure, which is also used to prove the Hardy-Littlewood maximal inequality. In the following, `mu` is the Lebesgue measure. We give little detail about the proofs in this section because proof scripts are essentially textbook, they can be found in MATHCOMP-ANALYSIS [1, file `lebesgue_measure.v`].

6.1 Egorov's theorem

Egorov's theorem relates convergence a.e. and uniform convergence (Sect. 5.2). Let A be a bounded measurable set, f_k be a sequence of functions measurable over A, and g be a function measurable over A. Suppose that f_k convergences a.e. relatively to the Lebesgue measure μ towards g. Then for any $\varepsilon > 0$, there exists a measurable set B such that $\mu(B) < \varepsilon$ and f_k converges uniformly towards g over $A \setminus B$. Formally, assuming `T` is a measurable type:

```
Lemma ae_pointwise_almost_uniform (f_ : (T -> R)^nat) (g : T -> R) A eps :
  (forall k, measurable_fun A (f_ k)) -> measurable_fun A g ->
  measurable A -> mu A < +oo ->
  {ae mu, forall x, A x -> f_ ^~ x @ \oo --> g x} ->
  (0 < eps)%R -> exists B, [/\ measurable B, mu B < eps%:E &
    {uniform A \ B, f_ @ \oo --> g}].
```

The notation for uniform convergence has been explained in Sect. 5.2.

6.2 Regularity

Proving the outer regularity of the Lebesgue measure is a preliminary step before proving its inner regularity.

6.2.1 Outer regularity

The Lebesgue measure is *outer regular*, which means that it can be approximated from above by open subsets. More formally, for every bounded measurable set D and $\varepsilon > 0$, there exists an open $U \supseteq D$ such that $\mu(U \setminus D) < \varepsilon$:

```
Lemma lebesgue_regularity_outer D eps :
  measurable D -> mu D < +oo -> (0 < eps)%R ->
  exists U : set R, [/\ open U , D ⊆ U & mu (U \ D) < eps%:E].
```

The proof is based on the definition of the Lebesgue measure as the infimum of the measures of covers, i.e., $\mu(X) = \inf_F \{\sum_{k=0}^{\infty} \mu(F_k) \,|\, (\forall k, \texttt{measurable}(F_k)) \wedge X \subseteq \bigcup_k F_k\}$.

6.2.2 Inner regularity

Intuitively, a measure is *inner regular* when it can be approximated from within by a compact subset. The inner regularity of the Lebesgue measure states that for every (bounded) measurable set D and $\varepsilon > 0$, there exists a compact set $V \subseteq D$ such that $\mu(D \setminus V) < \varepsilon$:

```
Lemma lebesgue_regularity_inner D eps :
  measurable D -> mu D < +oo -> (0 < eps)%R ->
  exists V : set R, [/\ compact V , V ⊆ D & mu (D \ V) < eps%:E].
```

Textbooks also resort to an alternative statement of inner regularity. Precisely, the above statement about a bounded measurable set can be generalized using the σ-finiteness of the Lebesgue measure by saying that the measure of a measurable set can be expressed as the supremum of the measure of the compact sets included inside, i.e., $\mu(D) = \sup\{\mu(K) \,|\, \mathrm{compact}(K) \wedge K \subseteq D\}$:

```
Lemma lebesgue_regularity_inner_sup D : measurable D ->
  mu D = ereal_sup [set mu K | K in [set K | compact K /\ K ⊆ D]].
```

As a matter of fact, we do use both forms in our development (Fig. 1).

7 A new proof of Urysohn's lemma

The classical version of Urysohn's lemma states that a topological space T is *normal* (i.e., two disjoint closed sets have disjoint open neighborhoods) if and only if, for any closed, disjoint, non-empty subsets A and B, there is a continuous function $f : T \to \mathbb{R}$ such that $f(A) = \{0\}$ and $f(B) = \{1\}$. This result is important because it connects a purely topological property (normality) with a purely analytic property (a function into the reals). Traditionally the proof involves an induction over the rationals to explicitly construct such a function. We do not follow the traditional proof for a technique that, we believe, is more appealing from the viewpoint of formal verification. Our proof has the same pattern as proving the Lebesgue Differentiation theorem first, and then the FTC: we found that proving the right intermediate lemmas made several results, including Urysohn's much easier.

7.1 Urysohn's lemma using uniform separator

We start by stating one of our intermediate lemmas. We first introduce a new definition. For a topological space `T`, we define a *uniform separator* as follows:

```
Definition uniform_separator (A B : set T) :=
  exists (uT : @Uniform.axioms_ T^o) (E : set (T * T)),
    let UT := Uniform.Pack uT in [/\
      @entourage UT E,
      (A `*` B) `&` E = set0 &
      (forall x, @nbhs UT UT x `<=` @nbhs T T x)].
```

This says that there is a uniform structure `UT` on `T` which separates `A` and `B`, and is coarser than the `T` topology. This is subtly different than assuming that `T` is a uniform space, which would imply `forall x, @nbhs UT UT x = @nbhs T T x`, which is too strong for our purposes. Also, `(A `*` B) `&` E = set0` has a nice visual: since an entourage is a region around the diagonal of `T * T`, `(A `*` B) `&` E = set0` means that the region `A `*` B` is far from the diagonal.

The key result about uniform separators is:

```
Lemma uniform_separatorP {T : topologicalType} {R : realType} (A B : set T) :
  uniform_separator A B <-> exists f : T -> R, [/\
    continuous f,
    range f `<=` `[0, 1],
    f @` A `<=` [set 0] &
    f @` B `<=` [set 1]].
```

For the sake of readability, we defer the explanation of the proof to the next section (Sect. 7.2).

The lemma just above is nearly Urysohn's lemma, but does not assume that `T` is a normal space. In fact, Urysohn's lemma follows immediately from the following lemma [1, file `normedtype.v`]:

```
Lemma normal_uniform_separator {T : topologicalType} (A : set T) (B : set T) :
  normal_space T -> closed A -> closed B -> A `&` B = set0 ->
  uniform_separator A B.
```

The advantage of the general lemma `uniform_separatorP` is that, besides Urysohn's lemma, we can derive other results. A *completely regular space* T is a topological space where for every point x and closed set A with $x \notin A$ there is a continuous function $f : T \to \mathbb{R}$

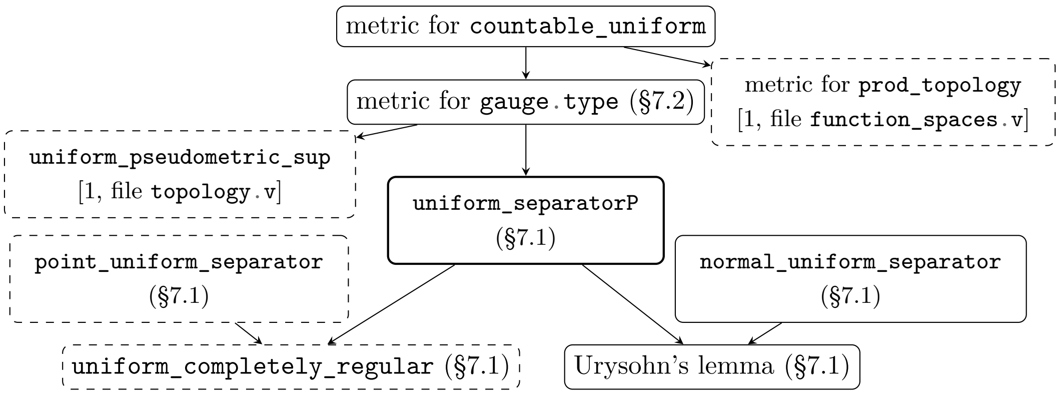
metric for countable_uniform
metric for gauge.type (§7.2)
metric for prod_topology [1, file function_spaces.v]
uniform_pseudometric_sup [1, file topology.v]
uniform_separatorP (§7.1)
point_uniform_separator (§7.1)
normal_uniform_separator (§7.1)
uniform_completely_regular (§7.1)
Urysohn's lemma (§7.1)

Figure 2 Overview of a novel proof of Urysohn's lemma and derived results.
(dashed nodes are derived results that are not relevant to the Lebesgue Differentiation theorem)

with $f(x) = 0$ and $f(A) = \{1\}$. The classical result is that uniform spaces are completely regular (lemma `uniform_completely_regular` in [1, file `normedtype.v`]). This follows from the lemma `uniform_separatorP` and the following lemma:

```
Lemma point_uniform_separator {uniformType T} (x : T) (B : set T) :
  closed B -> ~ B x -> uniform_separator [set x] B.
```

So, just like this paper formalizes the Lebesgue Differentiation theorem and proves the FTC among others, we find out that the lemma `uniform_sepratorP` proves Urysohn's lemma and more.

7.2 Existence of uniform separators

The proof of `uniform_separatorP` from the previous section (Sect. 7.1) follows other intermediate lemmas. If T is a uniform space with a countable basis for its uniformity (i.e., the set of entourages of T is the upward closure of a countable subset of entourages, see `countable_uniformity` [1, file `topology.v`]), then T has a pseudometric [15]. The construction of the metric is rather involved, but is only done once. Then, whenever we want a function into the reals, we construct a suitable uniformity instead, and rely on this result to guarantee such a function. This has several useful consequences, such as proving countable products of metric spaces are metrizable (Fig. 2). More importantly, the metrizability of uniform spaces lets us build the so-called *gauge metric*. Given an entourage E, we get a metric for the uniform structure generated by $E_0 = E \cap E^{-1}$, $E_1 = \texttt{split\_ent}(E_0) \cap \texttt{split\_ent}(E_0)^{-1}$, etc., where `split_ent` was explained in Sect. 2.

The direct part of the proof of `uniform_separatorP` starts by building the gauge metric of the separator of A and B generated from E. Since the initial entourage separates A and B, we know that there is an $\varepsilon > 0$ such that for all $x \in A$ and $y \in B$, $y \notin \mathrm{B}_\varepsilon(x)$. We define the extended real-valued function $d(x, y) \overset{\text{def}}{=} \inf\{r > 0 \,|\, y \in \mathrm{B}_r(x)\}$ (`edist` in [1, file `normedtype.v`]) and the extended real-valued function $d'(A, z) \overset{\text{def}}{=} \inf\{d(z, a) \,|\, a \in A\}$ (`edist_inf` in [1, file `normedtype.v`]). The function d is continuous on the gauge uniform space. Since it is coarser than the topology on T, d is continuous on $T \times T$, and thus d' is continuous on T. It follows that $f(x) \overset{\text{def}}{=} \min(d'(A, x), \varepsilon)/\varepsilon$ is continuous. It also takes 0 on A, and 1 on B, and has range $[0, 1]$, which means that we have the separating function and thus a proof of `uniform_separatorP`. See lemma `urysohn_separation` in [1, file `normedtype.v`] for details.

We now need to show `normal_uniform_separator`. For that purpose, we need to build a suitable uniform structure on a normal space. The uniformity we construct has the following basis:

```
Let apxU (UV : set T * set T) : set (T * T) :=
  (UV.2 × UV.2) ∪ ((closure UV.1)^C × (closure UV.1)^C).
Let nested (UV : set T * set T) :=
  [/\ open UV.1, open UV.2, A ⊆ UV.1 & closure UV.1 ⊆ UV.2].
Let ury_base := [set apxU UV | UV in nested].
```

Most of the work is to show that this is a basis for a uniformity, and that it is coarser than the topology on T. Then given an A and a B which are closed, disjoint, and non-empty, normality guarantees an open set $U \supseteq A$ with $U \cap B = \emptyset$. Then `apxU (U, B^C)` is a separator for A and B and we have our `uniform_separator`. See [1, file `normedtype.v`] for details.

8 Lusin and Tietze theorems and continuous functions are dense in L^1

As sketched in Sect. 4, one important step to prove the Lebesgue Differentiation theorem is to establish that continuous functions are dense in L^1. As we explained in Sect. 4, we can get along with a formal proof considering only functions defined over a bounded set. As an intermediate step, we formalize Lusin's theorem and Tietze's extension theorem.

8.1 Lusin's theorem

We place ourselves in a context with `R : realType` to represent a type of reals and where `mu` is the Lebesgue measure. Lusin's theorem states that, given a measurable function f over A (a measurable bounded set) and $\varepsilon > 0$, there exists a compact $K \subseteq A$ such that $\mu(K \setminus A) < \varepsilon$ and f is continuous within K [1, file `lebesgue_integral.v`]:

```
Lemma measurable_almost_continuous (f : R -> R) (A : set R) (eps : R) :
  measurable A -> mu A < +oo -> measurable_fun A f ->
  0 < eps -> exists K,
    [/\ compact K, K ⊆ A, mu (A \ K) < eps%:E & {within K, continuous f}].
```

The notation `{within _, continuous _}` was explained along the formalization of subspace topologies in Sect. 5.1. The proof also uses the following lemma that pertains to subspace topologies as well: if `f` and `g` are equal on `A` and `f` is continuous then so is `g`. Put formally:

```
Lemma subspace_eq_continuous {S : topologicalType} (f g : subspace A -> S) :
  {in A, f =1 g} -> continuous f -> continuous g.
```

The proof connects to results presented earlier in this paper: Egorov's theorem (Sect. 6.1) and the (bounded version of the) inner regularity of Lebesgue measurable (Sect. 6.2.2).

8.2 Tietze's extension theorem

Tietze's extension theorem states that in a normal topological space (normality being already defined in Sect. 7), a bounded, continuous, real-valued function on a closed set can be extended to a bounded, continuous function on the whole set. Although we do formalize Tietze's theorem for normal spaces, it should be noted that the normality condition is incidental to the main results of this paper; what is relevant here is that the reals are normal. Here follows the statement of Tietze's theorem in MathComp-Analysis [1, file `numfun.v`]:

```
Context {X : topologicalType} {R : realType}.
Hypothesis normalX : normal_space X.
Lemma continuous_bounded_extension (f : X -> R^o) (A : set X) M :
  closed A -> {within A, continuous f} ->
  0 < M -> (forall x, A x -> `|f x| <= M) ->
  exists g, [/\ {in A, f =1 g}, continuous g & forall x, `|g x| <= M].
```

The notation `^o` is only to help type-checking. Besides Urysohn's lemma, this proof uses the fact that uniform convergence preserves continuity (lemma `uniform_limit_continuous` in [1, file `function_spaces.v`]). The hypothesis `{within A, continuous f}` is a typical detail that does not appear in a textbook where this theorem would be assuming that the function `f` is continuous on `A`.

8.3 Continuous functions are dense in L^1

Finally, we arrive at the true goal of this section: the fact that continuous functions are dense in L^1, i.e., that given a function f integrable over a measurable, bounded set A, there exists a sequence of continuous functions g_k, integrable over A, such that $||f - g_k||_1$ tends towards 0:

```
Lemma approximation_continuous_integrable (A : set _) (f : R -> R):
  measurable A -> mu A < +oo -> mu.-integrable A (EFin \o f) ->
  exists g_ : (R -> R)^nat,
    [/\ forall n, continuous (g_ n),
        forall n, mu.-integrable A (EFin \o g_ n) &
        \int[mu]_(z in A) `|(f z - g_ n z)%:E| @[n --> \oo] --> 0].
```

The proof uses Tietze's and Lusin's theorems, see [1, file `lebesgue_integral.v`]. As we explained in Sect. 4, we use the above lemma to produce a sequence of continuous functions g_i to be used in the "bounded version" of the Lebesgue Differentiation theorem for a real-valued function restricted to some ball B_k. The desired sequence of g_i's is obtained by the above lemma modulo the technical detail that we need to restrict them to B_k for them to connect correctly to other lemmas used in the proof of the Lebesgue Differentiation theorem.

9 Covering lemmas and the Hardy-Littlewood maximal inequality

As we explained in Sect. 4, the second important step to prove the Lebesgue Differentiation theorem is the Hardy-Littlewood maximal inequality, i.e., the fact that, for all locally integrable functions f, $\mu(\{x \,|\, \mathrm{HL}(f)(x) > c\}) \leq \frac{3}{c}||f||_1$ for all $c > 0$. Its proof relies on a covering lemma typical of measure theory.

9.1 Vitali's covering lemma

In its finite version, the Vitali covering lemma can be stated as follows: given a finite collection of balls B_i with $i \in s$, there exists a subcollection B_j with $j \in D$ of pairwise disjoint balls such that $\bigcup_{i \in s} B_i \subseteq \bigcup_{j \in D} 3B_j$. To formalize this statement without committing to a concrete representation for collection of balls, we represent them by a function `B : I -> set R` such that each set satisfies a predicate `is_ball`, instead of representing them, say, as a function returning pairs of a center and a radius. The approach using the `is_ball` predicate gives rise to two functions `cpoint` and `radius` returning respectively a center point and a non-negative radius, when the set is indeed a ball. The finiteness of the collections is captured by using lists (respectively `s` and `D` below).

```
Context {I : eqType}.
Variable (B : I -> set R).
Hypothesis is_ballB : forall i, is_ball (B i).
Hypothesis B_set0 : forall i, B i !=set0.

Lemma vitali_lemma_finite (s : seq I) : { D : seq I | [/\ uniq D,
  {subset D <= s}, trivIset [set` D] B &
  forall i, i \in s -> exists j, [/\ j \in D, B i ∩ B j !=set0,
    radius (B j) >= radius (B i) & B i ⊆ 3 *` (B j)] ] }.
```

The notation `*`` represents scaling of the radius of a ball, i.e., `k *` B` is the open ball with center `cpoint B` and radius `k * radius B`.

We also formalized the infinite version of Vitali's covering lemma [1, file `normedtype.v`] and Vitali's theorem [1, file `lebesgue_measure.v`], which are much more involved. We did not need them to prove the Lebesgue Differentiation theorem but they served as a test-bed for using the `is_ball` predicate and are anyway often mentioned in connection with proofs of the FTC.

9.2 Hardy-Littlewood maximal inequality

The Hardy-Littlewood operator is a function that transforms a real-valued function f into the function

$$\mathrm{HL}(f)(x) \stackrel{\text{def}}{=} \sup_{r>0} \frac{1}{\mu(\mathrm{B}_r(x))} \int_{y \in \mathrm{B}_r(x)} |f(y)|(\mathbf{d}\,\mu).$$

Its formal definition uses elements similar to the ones used when defining Lebesgue points in Sect. 3:

```
Definition HL_max (f : R -> R) (x : R^o) (r : R) : \bar R :=
  (fine (mu (ball x r)))^-1%:E * \int[mu]_(y in ball x r) `|(f y)%:E|.
Definition HL_maximal (f : R -> R) (x : R^o) : \bar R :=
  ereal_sup [set HL_max f x r | r in `]0, +oo[ ].
```

The statement of the Hardy-Littlewood maximal inequality that we explained informally at the very beginning of this section (Sect. 9) then translates directly:

```
Lemma maximal_inequality (f : R -> R) c :
  locally_integrable [set: R] f -> 0 < c ->
  mu [set x | HL_maximal f x > c%:E] <= (3%:R / c)%:E * norm1 [set: R] f.
```

The L^1 norm is formalized in the obvious way as the identifier `norm1`. The proof relies on inner regularity (Sect. 6.2.2) and Vitali's covering lemma (Sect. 9.1). To establish that the Hardy-Littlewood operator is measurable, we also need to develop a theory of lower semicontinuity, which has been added to MathComp-Analysis on this occasion, see [1] for details.

10 Applications of the Lebesgue Differentiation theorem

In the previous sections, we have explained the main lemmas (mainly: continuous functions are dense in L^1 and the Hardy-Littlewood maximal inequality) used to prove the Lebesgue Differentiation theorem that we sketched in Sect. 4. We are now ready to proceed to direct applications.

10.1 The first FTC for Lebesgue integration

Recall from Sect. 1 the informal statement of the first FTC for Lebesgue integration: for $f \in L^1$, $F(x) \overset{\text{def}}{=} \int_{t \in]-\infty, x]} f(t)(\mathbf{d}\,\mu)$ is differentiable and $F'(x) \overset{\text{a.e.}}{=} f(x)$. This can furthermore be generalized to intervals of the form $]a, x]$ and $[a, x]$ and stated as a single theorem as follows [1, file `ftc.v`]:

```
Lemma FTC1_lebesgue_pt f a : mu.-integrable [set: R] (EFin \o f) ->
  let F x := (\int[mu]_(t in [set` Interval a (BRight x)]) (f t))%R in
  forall x, a < BRight x -> lebesgue_pt f x ->
  derivable F x 1 /\ F^`() x = f x.
```

The variable `a` has the generic type of an "interval bound" and `BRight` stands for closed bounds on the right [11, file `interval.v`]. The predicate `derivable` is for derivability (`1` is the direction) and the notation `^`()` is for derivatives with domain $\mathbb{R}$ [3, §4.5]. The theorem above connects these notions with the Lebesgue integral developed in MathComp-Analysis independently [2, §6.4]. The proof is standard in that it goes through a generalization of the Lebesgue Differentiation theorem where balls are replaced with *nicely shrinking* sets [16, §II.4.1]. The statement of the first FTC from Sect. 1 is an immediate corollary of the above lemma:

```
Corollary FTC1Ny f : mu.-integrable setT (EFin \o f) ->
  let F x := (\int[mu]_(t in [set` `]-oo, x]]) (f t))%R in
  {ae mu, forall x, derivable F x 1 /\ F^`() x = f x}.
```

10.2 Lebesgue's density theorem

Lebesgue's density theorem is another direct consequence of the Lebesgue Differentiation theorem. The *density* of a point x w.r.t. a set A is defined by $\lim_{r \to 0^+} \frac{\mu(A \cap \mathrm{B}_r(x))}{\mu(\mathrm{B}_r(x))}$. Lebesgue's density theorem states that almost everywhere the density is 0 or 1:

```
Lemma density (A : set R) : measurable A ->
  {ae mu, forall x, mu (A ∩ ball x r) * (fine (mu (ball x r)))^-1%:E
    @[r --> 0^'+] --> (\1_A x)%:E}.
```

11 Related work

We have been using various documents to formalize the Lebesgue Differentiation theorem. In particular, the main lines are drawn from lecture notes by Bowen [8]. For the proofs of the Hardy-Littlewood maximal inequality and the proof of the Lebesgue Differentiation theorem, we used books by Li [16] and Schwartz [28]. Surely, the same contents can be found elsewhere.

Several lemmas that we discussed can also be found in Mathlib [30]. Of course, the proof of Urysohn's lemma in Mathlib [23] is different from ours, which is original, as we explained in Sect. 7. Tietze's extension theorem in Mathlib [22, class `TietzeExtension`] has a similar statement and a similar proof. The statement of the Lebesgue Differentiation theorem in Mathlib [19] is more general than ours: it allows the domain of the function to be an arbitrary metric space, the measure can be any locally finite measure, the codomain can be any normed abelian group, and the balls (used in `davg` in Sect. 3) can be replaced by an arbitrary Vitali family. The Lebesgue Differentiation theorem in Mathlib is also used to prove a generic version of Lebesgue's density theorem [18] [26, §3.2].

Table 1 Estimated lines of code for the formalization of the Lebesgue Differentiation theorem and its direct applications.
(the column l.o.c. contains the number of lines of code in proof scripts for the main proof and intermediate lemmas (including statements); these numbers are approximations because, among other reasons, where one draws the line between intermediate lemmas and the supporting theories can be arbitrary)

Supporting theories	Section	l.o.c.		file in [1]
Subspaces	Sect. 5.1	N.A.		`topology.v`
Uniform convergence	Sect. 5.2	N.A.		`function_spaces.v`
Main lemmas	Section	l.o.c.		file in [1]
Egorov's thm	Sect. 6.1	$\approx$ 87	(2 lemmas)	`lebesgue_measure.v`
Outer regularity	Sect. 6.2.1	$\approx$ 61	(1 lemma)	`lebesgue_measure.v`
Inner regularity	Sect. 6.2.2	$\approx$ 118	(4 lemmas)	`lebesgue_measure.v`
Lusin's thm	Sect. 8.1	$\approx$ 108	(3 lemmas)	`lebesgue_integral.v`
Tietze's extension thm	Sect. 8.2	$\approx$ 108	(3 lemmas)	`numfun.v`
Density of cont. functions	Sect. 8.3	$\approx$ 118	(3 lemmas)	`lebesgue_integral.v`
Finite Vitali's covering lem.	Sect. 9.1	$\approx$ 75	(2 lemmas)	`normedtype.v`
Hardy-Littlewood max. ineq.	Sect. 9.2	$\approx$ 180	(6 lemmas)	`lebesgue_integral.v`
Urysohn's lemma	Sect. 7.1	$\approx$ 165	(11 lemmas)	`normedtype.v`
Lebesgue Differentiation thm	Sect. 4	$\approx$ 143	(3 lemmas)	`lebesgue_integral.v`
First FTC	Sect. 10.1	$\approx$ 265	(4 lemmas)	`ftc.v`
Lebesgue's density thm	Sect. 10.2	$\approx$ 69	(1 lemma)	`lebesgue_integral.v`
	Total (*Main lemmas*)	$\approx$ 1,497		

The FTC has already been the target of several formalizations in proof assistants. It can be found in Coq but for the Riemann integral in a constructive setting [10, §6]. NASAlib does not feature the first FTC for Lebesgue integration but an elementary version (for a C^1 function) of the second FTC [25, file `lebesgue_fundamental.pvs`], which can actually be obtained from the first FTC for Lebesgue integration as a corollary. Isabelle/HOL features the first FTC for Lebesgue integration but for continuous functions whereas we prove it for integrable functions [6, §3.7]. Mathlib features several variants of the first FTC; many require integrability and continuity at the endpoints but establish strict differentiability [20]. They stem from a lemma analogous to a strengthening of the Lebesgue Differentiation theorem with nicely shrinking sets [21]. In other words, we are able to match our lemmas with Mathlib lemmas but statements and proofs are organized in a different way. However, it must be said that Mathlib's statements are admittedly more general than ours in many respects. One reason is that MathComp-Analysis has started to use Hierarchy-Builder pervasively only recently. Before that, mathematical structures were manually encoded with *packed classes* [12]: this was making modifications very difficult in practice. With Hierarchy-Builder, we believe that introduction of, say, Banach spaces, should be a matter of engineering because most of our proofs are textbook, because we do not abuse the fact that we are working on the real line, and because our development is short enough to be refactored (see Table 1 for a concrete size estimation).

12 Conclusions

In this paper, we provided a comprehensive overview of the Lebesgue Differentiation theorem. We started with a formal statement and a proof overview (in Sect. 3 and in Sect. 4) to plan the whole development (as summarized in Fig. 1). Before being able to even state the first intermediate lemmas, we needed to extend MathComp-Analysis with in particular new topological constructs in Sect. 5. This made it possible to formalize the needed basic lemmas from measure theory in Sect. 6. Among the needed lemma, we formalized in particular a novel proof of Urysohn's lemma in Sect. 7. We used all this material to formalize the main steps of the proof of the Lebesgue Differentiation theorem: the density of continuous functions in L^1 in Sect. 8 and Hardy-Littlewood maximal inequality in Sect. 9. Our formalization of the Lebesgue Differentiation theorem was completed by two applications in Sect. 10, which include the first proof of the first FTC for Lebesgue integration for the Coq proof assistant.

In the end, we provide in a single document a complete overview of an important theorem. We believe that this experiment also concretely illustrates an important aspect of formalization of mathematics: the Lebesgue Differentiation theorem, like the uniform separators of Sect. 7.1, are examples of results whose formalization should be prioritized because, though technical, they are generic intermediate results from which important results can be obtained as corollaries (here, the first FTC and Urysohn's lemma). More pragmatically, we hope that this overview also contributes to documenting formalization of real analysis with MathComp-Analysis, for example by explaining the use of Hierarchy-Builder to develop topology. We think that we demonstrated that MathComp-Analysis is already a rich library and also a useful tool to formalize mathematics. As a matter of fact, we could use it to revisit Urysohn's lemma by producing an original proof.

As for future work, we are now working on the second FTC for Lebesgue integration whose most general form deals with *absolutely continuous functions* [24], using as the main ingredient the Radon-Nikodým theorem already available in MathComp-Analysis [14] and the recently developed theory of bounded and total variation [1, file `realfun.v`].

References

1 Reynald Affeldt, Yves Bertot, Alessandro Bruni, Cyril Cohen, Marie Kerjean, Assia Mahboubi, Damien Rouhling, Pierre Roux, Kazuhiko Sakaguchi, Zachary Stone, Pierre-Yves Strub, and Laurent Théry. MathComp-Analysis: Mathematical Components compliant analysis library. `https://github.com/math-comp/analysis`, 2017. Last stable version: 1.2.0 (2024).

2 Reynald Affeldt and Cyril Cohen. Measure construction by extension in dependent type theory with application to integration. *J. Autom. Reason.*, 67(3):28:1–28:27, 2023.

3 Reynald Affeldt, Cyril Cohen, and Damien Rouhling. Formalization techniques for asymptotic reasoning in classical analysis. *J. Formaliz. Reason.*, 11(1):43–76, 2018. `doi:10.6092/issn.1972-5787/8124`.

4 Reynald Affeldt, Cyril Cohen, and Ayumu Saito. Semantics of probabilistic programs using s-finite kernels in Coq. In *12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2023), Boston, MA, USA, January 16–17, 2023*, pages 3–16. ACM, 2023. `doi:10.1145/3573105.3575691`.

5 Reynald Affeldt and Zachary Stone. Towards the fundamental theorem of calculus for the Lebesgue integral in Coq. In *35ème Journées Francophones des Langages Applicatifs (JFLA 2024), Saint-Jacut-de-la-Mer, France, January 30–February 2*, January 2024. open access.

6 Jeremy Avigad, Johannes Hölzl, and Luke Serafin. A formally verified proof of the central limit theorem. *J. Autom. Reason.*, 59(4):389–423, 2017. `doi:10.1007/s10817-017-9404-x`.

7 Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond. Coquelicot: A user-friendly library of real analysis for Coq. *Math. Comput. Sci.*, 9(1):41–62, 2015. `doi:10.1007/S11786-014-0181-1`.

8 Lewis Bowen. Lecture notes in real analysis. Available at `https://web.ma.utexas.edu/users/lpbowen/m381c/lecture-notes.pdf`, December 2014. University of Texas at Austin.

9 Cyril Cohen, Kazuhiko Sakaguchi, and Enrico Tassi. Hierarchy builder: Algebraic hierarchies made easy in Coq with Elpi (system description). In *5th International Conference on Formal*

Structures for Computation and Deduction (FSCD 2020), June 29–July 6, 2020, Paris, France (Virtual Conference), volume 167 of *LIPIcs*, pages 34:1–34:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.FSCD.2020.34.

10 Luís Cruz-Filipe. A constructive formalization of the fundamental theorem of calculus. In *Selected Papers of the 2nd International Workshop on Types for Proofs and Programs (TYPES 2002), Berg en Dal, The Netherlands, April 24–28, 2002*, volume 2646 of *Lecture Notes in Computer Science*, pages 108–126. Springer, 2002. doi:10.1007/3-540-39185-1_7.

11 The MathComp development team. Mathematical components. https://github.com/math-comp/math-comp, 2005. Last stable version: 2.2.0 (2024).

12 François Garillot, Georges Gonthier, Assia Mahboubi, and Laurence Rideau. Packaging mathematical structures. In *22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009), Munich, Germany, August 17–20, 2009*, volume 5674 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2009. doi:10.1007/978-3-642-03359-9_23.

13 Johannes Hölzl, Fabian Immler, and Brian Huffman. Type classes and filters for mathematical analysis in Isabelle/HOL. In *4th International Conference on Interactive Theorem Proving (ITP 2013), Rennes, France, July 22–26, 2013*, volume 7998 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2013. doi:10.1007/978-3-642-39634-2_21.

14 Yoshihiro Ishiguro and Reynald Affeldt. The Radon-Nikodým theorem and the Lebesgue-Stieltjes measure in Coq. *Computer Software*, 41(2), 2024. Japan Society for Software Science and Technology.

15 Samuel Leland Lesseig. Metrization of uniform spaces. Master's thesis, Department of Mathematics, Kansas State University, Manhattan, Kansas, 1963.

16 Daniel Li. *Notions fondamentales d'analyse réelle et complexe—Espaces de Hardy et interpolation, avec exercices corrigés*. Ellipses, 2022.

17 Assia Mahboubi and Enrico Tassi. *Mathematical Components*. Zenodo, January 2021. doi:10.5281/zenodo.4457887.

18 Mathlib 4. File MeasureTheory/Covering/DensityTheorem.lean. url, June 2024.

19 Mathlib 4. File MeasureTheory/Covering/Differentiation.lean. url, June 2024.

20 Mathlib 4. File MeasureTheory/Integral/FundThmCalculus.lean. url, June 2024.

21 Mathlib 4. File MeasureTheory/Integral/SetIntegral.lean. url, June 2024.

22 Mathlib 4. File Topology/TietzeExtension.lean. url, June 2024.

23 Mathlib 4. File Topology/UrysohnsLemma.lean. url, June 2024.

24 Maran Mohanarangan. The fundamental theorem of calculus for Lebesgue integration. Technical report, ETH Zürich, 2021. Exercise Class 13, Measure and Integration, Spring 2021, available at https://metaphor.ethz.ch/x/2021/fs/401-2284-00L/sc/notes_exclass13.pdf.

25 NASALib. NASA PVS library of formal developments. Current version: 7.1.1. Available at https://github.com/nasa/pvslib, 2023.

26 Oliver Nash. A Formalisation of Gallagher's Ergodic Theorem. In *14th International Conference on Interactive Theorem Proving (ITP 2023)*, volume 268 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:16, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITP.2023.23.

27 Ayumu Saito and Reynald Affeldt. Experimenting with an intrinsically-typed probabilistic programming language in Coq. In *21st Asian Symposium on Programming Languages and Systems (APLAS 2023), Taipei, Taiwan, November 26–29, 2023*, volume 14405, pages 182–202. Springer, 2023. doi:10.1007/978-981-99-8311-7_9.

28 Laurent Schwartz. *Analyse III: Calcul intégral*. Hermann, 1997.

29 The Coq Development Team. *The Coq Proof Assistant Reference Manual*. Inria, 2024. Available at https://coq.inria.fr. Version 8.19.2.

30 The mathlib Community. The lean mathematical library. In *9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020), New Orleans, LA, USA, January 20–21, 2020*, pages 367–381. ACM, 2020. doi:10.1145/3372885.3373824.

31 Li Zhou, Gilles Barthe, Pierre-Yves Strub, Junyi Liu, and Mingsheng Ying. CoqQ: Foundational verification of quantum programs. *Proc. ACM Program. Lang.*, 7(POPL):833–865, 2023. `doi:10.1145/3571222`.