Mathematical insights from using Lean

Damiano Testa

University of Warwick

Lean Together 2021, January 7th, 2021

About myself

For those who do not know me, I consider myself an algebraic geometer, with a strong interest in number theory.

My computer experience prior to this summer was

- browsing the internet,
- typing in LaTeX,
- using the computational algebra system called Magma.

However, I know Kevin Buzzard and I was aware of his efforts in formalization of mathematics.

Disclaimer

I started using Lean over the summer and I am still a beginner.

I feel confident in the mathematical side of my presentation.

The Lean side is quite a bit shakier: I welcome all sorts of comments!

I am going to present my personal experience with my first, serious, ongoing formalization project.

In these slides, what is trivial, is formalized; what is interesting is in progress.

The ideas expressed in this presentation are mostly reflection of my limited understanding of how mathematics can be formalized.

Chevalley's Theorem

My long-term, yet unreached, goal is to formalize Chevalley's Theorem.

Theorem (Chevalley)

Let $\pi\colon X \to Y$ be a morphism of schemes [more assumptions]. The image of a constructible subset of X is constructible.

Do not worry if you do not understand the meaning of most of the words in this statement: we will reduce to polynomials very quickly!

As an algebraic geometer, I view this statement as an important and useful result.

Theorem (Chevalley)

Let $\pi\colon X \to Y$ be a morphism of schemes [more assumptions].
The image of a constructible subset of X is constructible.

Constructible means a finite union of intersections of open and a closed set.

I also thought that it was relatively easy to prove…on paper!

Reduction steps: it suffices to consider the case

Geometry side	Algebra side
$X = Y \times \mathbb{A}^r$ and $\pi\colon Y \times \mathbb{A}^r \to Y$ is the projection	R is a `comm_ring` ($\leftrightarrow Y$) study the inclusion $R \to R[x_1, \dots, x_r]$
$r = 1$ $\pi\colon Y \times \mathbb{A}^1 \to Y$	`{R : Type*} [comm_ring R]` `C : R → polynomial R`

Chevalley's Theorem – reductions

Lemma (Simplified Chevalley's Theorem, version 1)

Let $\pi \colon Y \times \mathbb{A}^1 \to Y$ *be the projection onto the first factor.*

Given U, V *open subsets of* $Y \times \mathbb{A}^1$, *the set* $\pi(U \cap V^c)$ *is constructible.*

Recall: constructible is a finite union of intersections of open and a closed set.

It turns out that the projection map $Y \times \mathbb{A}^1 \to Y$ is *open*, and we can break our goal further:

1. if $U \subset Y \times \mathbb{A}^1$ is open, then show that $\pi(U) \subset Y$ is open (instead of just constructible).
2. if $C \subset Y \times \mathbb{A}^1$ is closed, then show that $\pi(C) \subset Y$ is constructible.

Progress with formalization

Lemma (Simplified Chevalley's Theorem, version 2)

Let $\pi\colon Y \times \mathbb{A}^1 \to Y$ be the projection onto the first factor.

1. *The morphism π is open.*
2. *if $C \subset Y \times \mathbb{A}^1$ is closed, then $\pi(C) \subset Y$ is constructible.*

Item (1) is fully formalized! (More on this below.)

Item (2) is still in progress.

A proof that $\pi\colon Y \times \mathbb{A}^1 \to Y$ is open

```
import algebraic_geometry.prime_spectrum
import ring_theory.polynomial.basic
import tactic

open ideal polynomial prime_spectrum set

local attribute [instance] classical.prop_decidable

universe u

/- The morphism Spec R[x] --> Spec R induced by the natural inclusion R --> R[x] is an open map -/
section Rx2Ropen

/- quoisintdomain shows that if P is a prime ideal of R, then R[x]/(P) satisfies is_integral_domain -/
lemma quoisintdomain {R : Type u} [comm_ring R] {P : ideal R} (H : is_prime P) : is_integral_domain
(quotient (map C P : ideal (polynomial R))) :=
begin
  let quo := polynomial (quotient P),
  let quot := quotient (map C P : ideal (polynomial R)),
  have idq : is_integral_domain quo := integral_domain.to_is_integral_domain quo,
  have iso : quo ≃+* quot := polynomial_quotient_equiv_quotient_polynomial P,
  symmetry' at iso,
  exact ring_equiv.is_integral_domain quo idq iso,
end

/- liftprime shows that if P is a prime ideal of R, then P.R[x] is a prime ideal of R[x] -/
lemma liftprime {R : Type u} [comm_ring R] {P : ideal R} (H : is_prime P) : is_prime (map C P : ideal
(polynomial R)) :=
begin
  let I : ideal (polynomial R) := map C P,
  refine (quotient.is_integral_domain_iff_prime I).mp _,
  exact quoisintdomain H,
end

variables {R : Type u} [comm_ring R]
noncomputable def Cstar := prime_spectrum.comap (polynomial.C : R →+* polynomial R)

lemma incl {R : Type u} [comm_ring R] (I : prime_spectrum (polynomial R))
 : ∀ p ∈ (Cstar I).as_ideal , polynomial.C p ∈ I.as_ideal :=
begin
  intros p hp, exact hp,
end

def image_of_Df {R : Type u} [comm_ring R] (f : polynomial R) := {p : prime_spectrum R | ∃ i : ℕ ,
(coeff f i) ∉ p.1}


lemma std_aff_is_open {R : Type u} [comm_ring R] (a : R) : is_open ({p : prime_spectrum R | a ∉ p.1 }) :=
begin
  refine ⟨ {a} , _ ⟩,
```

```
  let Utot := {p : prime_spectrum R | ∃ i : ℕ , coeff f i ∉ p.1 },
  let Uis := {Ui : set (prime_spectrum R) | ∃ i : ℕ , Ui = {p : prime_spectrum R | coeff f i ∉ p.1 }},
  have uunion : Utot = ⋃₀ Uis,
  { ext1,
    split,
    { intro xh,
      simp at *,
      cases xh with i ih,
      use {p : prime_spectrum R | coeff f i ∉ p.1 },
      use i,
      refl, },
    { intro xh,
      simp at *,
      rcases xh with ⟨ Ui , i , xui ⟩ ,
      cases i with i ih,
      use i,
      rw ih at xui,
      exact xui, }, },
  have allopen : ∀ i : ℕ , is_open {p : prime_spectrum R | coeff f i ∉ p.1 },
  { intro i,
    exact std_aff_is_open (f.coeff i), },
  refine (is_open_iff {p : prime_spectrum R | ∃ (i : ℕ), f.coeff i ∉ p.val}).mpr _,
  use {r : R | ∃ i : ℕ , r = coeff f i},
  ext1,
  split,
  { intro xh,
    simp at *,
    intros co coh,
    have ox : ∃ (i : ℕ), co = f.coeff i,
    { exact coh, },
    cases ox with ii,
    rw ox_h,
    exact xh ii,
  },
  { intro xh,
    simp at *,
    intro j,
    have ech : ∀ i : ℕ , f.coeff i ∈ x.1,
    { intro iii,
      refine mem_def.mpr _,
      have : f.coeff iii ∈ {r : R | ∃ (i : ℕ), r = f.coeff i},
      { refine mem_def.mpr _,
        refine set_of_app_iff.mpr _,
        use iii, },
      exact xh this, },
    exact ech j },
end
```

A proof that $\pi\colon Y \times \mathbb{A}^1 \to Y$ is open – frame 2

```
    rw mul_comm,
    refine submodule.smul_mem _ _ _,
    apply submodule.subset_span,
    use i, },
  { have h2 : f.coeff i = 0, by simpa using h,
    simp [h2], }
end

/- this lemma shows that the image of Df is contained in an open set (with which it actually coincides)
-/
lemma imma_f2ind {R : Type u} [comm_ring R] (f : polynomial R) (I : prime_spectrum (polynomial R)) : f ∉ I.1 → ∃ i : ℕ , coeff f i ∉ (Cstar I).1 :=
begin
  let P := Cstar I,
  by_contradiction a,
  push_neg at a,
  cases a with fni cini,
  apply fni,
  have fins : f ∈ ideal.span {tt : polynomial R | ∃ i : ℕ ,tt = (C (coeff f i))}, by exact fincN f,
  have pas : ∀ (i : ℕ), f.coeff i ∈ P.1, by exact cini,
  have : ∀ p ∈ P.1 , C p ∈ I.1,
  { intros p hp,
    exact incl I p hp, },
  have anc : ∀ (i : ℕ), C (f.coeff i) ∈ I.val,
  { intro i,
    exact this (f.coeff i) (cini i), },
  have anc1 : {tt : polynomial R | ∃ (i : ℕ), tt = C (f.coeff i)} ⊆ I.val,
  { intros tt tth,
    cases tth with i hi,
    rw hi,
    exact (submodule.mem_coe I.val).mpr (this (coeff f i) (cini i)), },
  have anc3 : (span {tt : polynomial R | ∃ (i : ℕ), tt = C (f.coeff i)}).1 ≤ I.val,
  { have : (span {tt : polynomial R | ∃ (i : ℕ), tt = C (f.coeff i)}).1 ≤ (span ↑(I.val)).1,
    { apply span_mono,
      exact anc1, },
    exact bInter_subset_of_mem anc1, },
  { exact anc3 fins, },
end


/- imma_spec shows that if a point of Spec R[x] is not contained in the vanishing set of f,
-- then its image in Spec R is contained in the open where at least one of the coefficients
-- of f is non-zero. This lemma is a reformulation of imma_f2ind -/
lemma imma_spec {R : Type u} [comm_ring R] {f : polynomial R} {I : prime_spectrum (polynomial R)} (H : I ∈ (prime_spectrum.zero_locus {f} : set (prime_spectrum (polynomial R)))ᶜ ) : Cstar I ∈ image_of_Df f :=
begin
  simp * at *,
  apply imma_f2ind,
  exact H,
```

```
  apply hi,
  exact mem_map_C_iff.mp a i,
end



lemma updown {R : Type u} [comm_ring R] (r : R) : r = coeff (C r) 0 :=
begin
  exact coeff_C_zero.symm,
end


lemma updownPid {R : Type u} [comm_ring R] (r : R) {P : prime_spectrum R} {H : r ∈ P.1} : r ∈ ((ideal.comap (C : R →+* polynomial R)) ((map C P.1 : ideal (polynomial R)))) :=
begin
  let liftP := ideal.map (polynomial.C : R →+* polynomial R) P.1,
  have req : {r} = C⁻¹' (C '' {r}),
  { ext1,
    split,
    { intro hx,
      have : x=r, by exact hx,
      rw this,
      finish, },
    { intro hx,
      refine mem_singleton_iff.mpr _,
      simp * at *,
      exact hx, }, },
  { unfold ideal.map,
    apply submodule.subset_span,
    exact mem_image_of_mem C.to_fun H, },
  exact ring.to_semiring,
  exact set.has_singleton,
end


lemma liftproj {R : Type u} [comm_ring R] (P : prime_spectrum R) : (prime_spectrum.comap (polynomial.C : R →+* polynomial R)) ⟨ (ideal.map C P.1 : ideal (polynomial R)) , by exact (quotient.is_integral_domain_iff_prime (ideal.map C P.val : ideal (polynomial R))).mp (quoisintdomain P.2) ⟩ = P :=
begin
  let lift : prime_spectrum (polynomial R) := ⟨ (ideal.map C P.1 : ideal (polynomial R)) , by exact (quotient.is_integral_domain_iff_prime (ideal.map C P.val : ideal (polynomial R))).mp (quoisintdomain P.2) ⟩,
  ext1,
  ext,
  split,
  { intro hx,
    rw updown x,
    exact mem_map_C_iff.mp hx 0, },
  { intro hx,
```

$\pi\colon Y \times \mathbb{A}^1 \to Y$ is open – goals accomplished

```
  intros U hU,
-- rewrite the openness condition using zero loci
  rw is_open_iff at hU,
-- fs are the polynomials whose vanishing set defines the complement of U (cl)
  cases hU with fs cl,
-- observe that the union of the singletons of the polynomials in fs is fs!
  have funi : fs = (⋃ i : fs , {i.1}),
    {ext1,
    split;finish},
-- use the observation on the union of singletons
  rw funi at cl,
-- use that the zero locus of a union is the intersection of the zero loci
  rw zero_locus_Union at cl,
-- take complements
  have uuni : U = (⋂ (i : ↥fs), zero_locus {i.val})ᶜ,
    {rw ← cl,
    symmetry,
    exact compl_compl U,},
-- the complement of the intersection is the union of the complements
  rw compl_Inter at uuni,
-- use that U is a union
  rw uuni,
-- the image of the union is the union of the images
  rw image_Union,
-- to show that the union is open, we show that each union-and is open
  apply is_open_Union,
-- call f one of the functions
  intro f,
-- we are going to show that the image of Spec R[x]_f is D(coefficients of f)
-- this should be a combination of surj, showing surjectivity, and of
-- imma_f2ind, showing that the image is contained in the given open set
  have image : (Cstar) '' (zero_locus {f.val})ᶜ = image_of_Df f.1,
  { ext1,
    split,
    { intro hx,
      cases hx with xli,
      cases hx_h with complement img,
      symmetry' at img,
      rw img,
      apply imma_spec,
      exact complement, },
    { intro hx,
      simp * at *,
      use (ideal.map C x.val : ideal (polynomial R)),
      { exact liftprime x.2, },
      { split,
        { exact surj ↑f x hx, },
        { exact liftproj x, }, }, }, },
  rw image,
-- let us show that the set image_of_Df f.1 is indeed open, observing that this is lemma total_image
  exact total_image f.1,
end

end Rx2Ropen
```

▼ Tactic state

goals accomplished 🎉

▶ All Messages (0)

Golfing the proof

After that, I became better at golfing!

I am learning golfing tricks thanks to the combined efforts of the many users of the Zulip chat.

I am incredibly grateful for the time that everyone on Zulip devotes to answering the questions that appear there!

Shortened version

```
import algebraic_geometry.prime_spectrum
import ring_theory.polynomial.basic

open ideal polynomial prime_spectrum set
/--
The morphism `Spec R[x] --> Spec R` induced by the natural inclusion `R --> R[x]`
is an open map.
-/

local attribute [instance] classical.prop_decidable

variables {R : Type*} [comm_ring R] {P : ideal R}

/-- `quoisintdomain` shows that if `P` is a prime ideal of `R`,
then `R[x]/(P)` satisfies is_integral_domain. -/
lemma quoisintdomain (H : is_prime P) :
  is_integral_domain (quotient (map C P : ideal (polynomial R))) :=
ring_equiv.is_integral_domain (polynomial (quotient P))
  (integral_domain.to_is_integral_domain (polynomial (quotient P)))
  (polynomial_quotient_equiv_quotient_polynomial P).symm

/-- `liftprime` shows that if `P` is a prime ideal of `R`,
then `P.R[x]` is a prime ideal of `R[x]`. -/
lemma liftprime (H : is_prime P) : is_prime (map C P : ideal (polynomial R)) :=
(quotient.is_integral_domain_iff_prime (map C P : ideal (polynomial R))).mp (quoisintdomain H)

/-- `image_of_Df` is the subset of `Spec R[x]` that we will prove is the image of
the non-vanishing set of `f`. -/
def image_of_Df (f : polynomial R) :=
  {p : prime_spectrum R | ∃ i : ℕ , (coeff f i) ∉ p.1}

lemma std_aff_is_open (a : R) : is_open ({p : prime_spectrum R | a ∉ p.1 }) :=
begin
  refine ⟨{a} , ext (λ x, ⟨λ xh, _,λ xh,  _⟩)⟩;
  simpa only [mem_zero_locus, not_not, mem_set_of_eq, singleton_subset_iff, mem_compl_eq] using xh,
end

lemma total_image (f : polynomial R) :
  is_open ({p : prime_spectrum R | ∃ i : ℕ , coeff f i ∉ p.1 }) :=
begin
  rw set_of_exists (λ i (x : prime_spectrum R), coeff f i ∉ x.val),
  exact is_open_Union (λ i, std_aff_is_open (coeff f i)),
end

lemma fincN (f : polynomial R) :
  f ∈ ideal.span {tt : polynomial R | ∃ i : ℕ ,tt = (C (coeff f i))} :=
begin
  conv_lhs {rw as_sum_support f},
  refine submodule.sum_mem _ (λ i hi, _),
  rw [← C_mul_X_pow_eq_monomial, mul_comm],
  refine submodule.smul_mem _ _ (submodule.subset_span ⟨i, rfl⟩),
end

/-- `imma_f2ind` shows that the image of `Df` is contained in an open set
(with which it actually coincides). -/
lemma imma_f2ind (f : polynomial R) (I : prime_spectrum (polynomial R)) :
  f ∉ I.1 → ∃ i : ℕ , coeff f i ∉ (prime_spectrum.comap (polynomial.C : R →+* polynomial R) I).1 :=
begin
  contrapose!,
  intros cini,
  have anc1 : {tt : polynomial R | ∃ (i : ℕ), tt = C (f.coeff i)} ⊆ I.val,
  { rintros tt ⟨i, rfl⟩,
    exact (submodule.mem_coe I.val).mpr (cini i) },
  have anc3 : (span {tt : polynomial R | ∃ (i : ℕ), tt = C (f.coeff i)}).1 ⊆ I.val :=
    bInter_subset_of_mem anc1,
  { refine (anc3) (fincN f), },
end

/-- `imma_spec` shows that if a point of `Spec R[x]` is not contained in the vanishing set of `f`,
then its image in `Spec R` is contained in the open where at least one of the coefficients
of `f` is non-zero. This lemma is a reformulation of `imma_f2ind`. -/
lemma imma_spec {f : polynomial R} {I : prime_spectrum (polynomial R)}
  (H : I ∈ (prime_spectrum.zero_locus {f} : set (prime_spectrum (polynomial R)))ᶜ ) :
  prime_spectrum.comap (polynomial.C : R →+* polynomial R) I ∈ image_of_Df f :=
begin
  refine imma_f2ind f I _,
  rw [mem_compl_eq, mem_zero_locus, singleton_subset_iff] at H,
  exact H,
end

theorem Rx2Ropen :
  is_open_map (prime_spectrum.comap (polynomial.C : R →+* polynomial R)) :=
begin
  rintros U ⟨fs, cl⟩,
  have funis : fs = (⋃ i : fs , {i.1}),
  { refine ext (λ x, ⟨λ hf, mem_Union.mpr (Exists.intro ⟨x, hf⟩ (mem_singleton x)), _⟩),
    rintro ⟨-, ⟨⟨-, xfs⟩, rfl⟩, ⟨⟩⟩,
    exact xfs },
  rw [funis, zero_locus_Union] at cl,
  have uuni : U = (⋂ (i : ↥fs), zero_locus {i.val})ᶜ,
  { obtain F := (congr_arg compl cl).symm,
    rwa compl_compl at F },
  rw [uuni, compl_Inter, image_Union],
  refine is_open_Union (λ f, _),
  convert total_image f.1,
  refine ext (λ x, ⟨_, λ hx, ⟨⟨map C x.1, (liftprime x.2)⟩, ⟨_, _⟩⟩⟩),
  { rintro ⟨xli, complement, rfl⟩,
    exact imma_spec complement },
  { rw [mem_compl_eq, mem_zero_locus, singleton_subset_iff],
    cases hx with i hi,
    exact λ a, hi (mem_map_C_iff.mp a i) },
  { refine subtype.ext (ext (λ x, ⟨λ h, _, λ h, submodule.subset_span (mem_image_of_mem C.1 h)⟩)),
    rw ← @coeff_C_zero _ x,
    exact mem_map_C_iff.mp h 0}
end
```

The proof went from approx 250 to approx 90 lines.

Of course, there is room for further compression.

Looking back at formalization

However, I have not inspected the proof with the idea of formalizing it.

I simply took *some* mathematical proof and converted it step by step.

The proofs above are based on Lemma 10.28.7 of the Stacks project and its dependencies.

The arguments in the Stacks project are very easy to read *for a human*.

For me, though, they were hard to formalize directly.
The end result is the initial long argument.

Golfing provided a layer of compression, basically navigating inside the same proof, but taking fewer detours.

Revising the proof with the idea of formalizing

Image credit: Robert Hodgin.

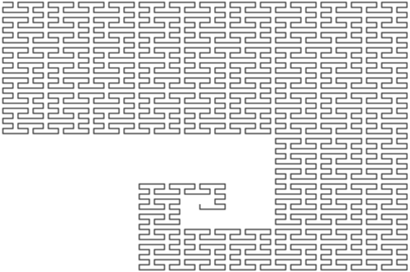

Image credit: Aubrey Jaffer.

Writing a mathematical paper gives a chance to revise proofs:

- many arguments are dead-ends,
- several lemmas go around in circles,
- some steps required a more details.

Revising the proof with the idea of formalizing

Image credit: Robert Hodgin.

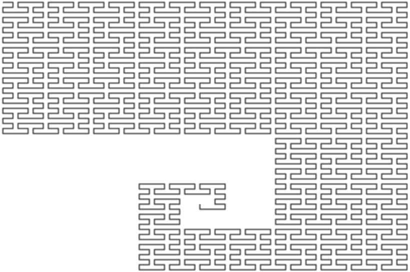

Image credit: Aubrey Jaffer.

For me, the initial plane-filling proof is almost a necessity.

I make sure that what I am saying is **true** by

- working out representative cases,
- thinking about extreme cases and pushing boundaries,
- removing hypotheses to see what fails,
- proving related, but unnecessary statements.

Plane-filling, meanders, API

I believe that the carpet of lemmas around a definition is related to what is called an API by the Lean community!

In the long run, the vast amount of auxiliary results, unneeded corollary, trivial implications, barely off-target proofs is what convinces me that what I want to prove is actually true.

After that, I write a proof.

Frame Title

I am confident that I am proving something true, **because** of the carpet of auxiliary results.

I am also confident that I may make mistakes in writing it down.

The meanderings around my arguments are the reason that I am confident that my mistakes are fixable.

What I am still getting used to, is that now **Lean** is making sure of the soundness of the arguments!

Of course, there is still the need of the carpet of lemmas of the API, but it now plays a somewhat different role, at least in my mind!

Chevalley's Theorem – formalization revised

Back to the projection map

$$\pi\colon \operatorname{Spec} R[x] \longrightarrow \operatorname{Spec} R$$

or, in Lean,

```
prime_spectrum.comap (C : R →+* polynomial R)
```

Going deeper into the proof, besides inductions, open covers, restrictions and tautologies, the main lemma is the following.

Lemma

Let R be a commutative ring and let $I \subset R$ be an ideal. A polynomial $f \in R[x]$ belongs to the ideal generated by I if and only if all the coefficients of f belong in I.

Lemma

Let R be a commutative ring and let $I \subset R$ be an ideal. A polynomial $f \in R[x]$ belongs to the ideal generated by I if and only if all the coefficients of f belong in I.

This lemma appears in `ring_theory/polynomial/basic.lean` under the name `mem_map_C_iff`.

Theorem (`mem_map_C_iff`)

```
{I : ideal R} {f :  polynomial R} :
f ∈ (ideal.map C I : ideal (polynomial R))  ↔
∀ n : ℕ, f.coeff n ∈ I :=
```

Also, I understand better the mathematical proof!

Future steps

I think that I can further substantially shorten the proof.

In the golfed proof above, lemma `mem_map_C_iff` only appears at the very end of the final proof.

I need to go back and rethink all the proofs with the aim of extending the application of `mem_map_C_iff`.

In fact, I might even discover that this is already developed in `mathlib`!

If someone knows that this is the case, then I would be extremely happy to learn this!

Thank you!

Questions?