Formalizing the solution to the cap set problem

Sander R. Dahmen, Johannes Hölzl, Robert Y. Lewis

Vrije Universiteit Amsterdam

CARMA Workshop on Computer-Aided Proof
Newcastle, NSW
June 6, 2019

Motivation

Formal mathematics

Proof assistants have seen lots of success in computer science applications.

Less in mathematics, outside of some noteworthy large-scale projects.

Across various systems: a good amount of undergraduate mathematics, a few major standalone projects.

Some problems:

- Most significant mathematical results rely on lots of background theory.
- Different theorems rely on different backgrounds, even when they come from the same subfields.
- Focusing on single theorems leads to irregular coverage of background theory.
- Automation needs to “keep pace” with the theory: different fields benefit from different kinds of proof search.

Lean Forward

A new project at the VU: formalize modern results in number theory, in Lean.

- Develop comprehensive libraries that will help with many results.
- Target "research areas"/collections of moderate difficulty results, instead of single challenge theorems.
- Work on the system and automation alongside the formalizing.
- PI: Jasmin Blanchette

Lean Forward

Can we formalize current results yet?

Sander Dahmen's first proposal: formalize Ellenberg and Gijswijt's solution to the cap set problem.

- Recent: *Annals of Mathematics*, 2017
- The theorem can be stated in elementary terms.
- The proof does not depend on any high-powered results, but…
- it uses a lot of elementary linear algebra: a good stress test.
- The "second half" of the proof can be made even more elementary.

Can we formalize current results yet? Yes! *

We have completed a proof of Ellenberg and Gijswijt's theorem in Lean.

- The first half of our proof is faithful to their argument.
- The second half takes a much more elementary approach.
- A lot of linear algebra, combinatorics, etc. was added to Lean's `mathlib`.
- We followed a detailed informal blueprint by Sander.

Paper and blueprint: `https://lean-forward.github.io/e-g/`

Can we formalize current results yet? Yes! *

We have completed a proof of Ellenberg and Gijswijt's theorem in Lean.

- The first half of our proof is faithful to their argument.
- The second half takes a much more elementary approach.
- A lot of linear algebra, combinatorics, etc. was added to Lean's `mathlib`.
- We followed a detailed informal blueprint by Sander.

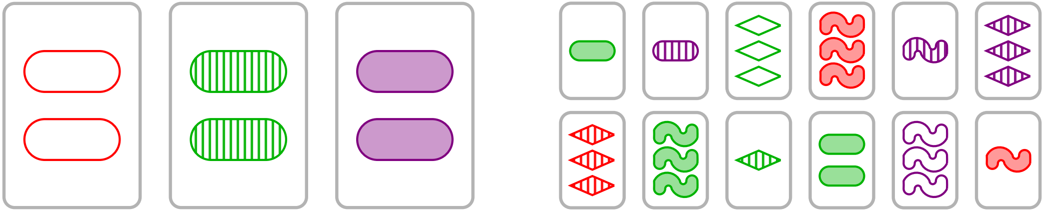
Paper and blueprint: `https://lean-forward.github.io/e-g/`

(*) This was a very special case.

Table of contents

The cap set problem

The cap set problem



The cap set problem

Specific statement

Let $r_3(G)$ denote the cardinality of a largest subset of an abelian group G containing no three-term arithmetic progression. Is there a constant $c < 3$ such that $r_3((\mathbb{Z}/3\mathbb{Z})^n)$ grows in n no faster than c^n?

The cap set problem

Specific statement

Let $r_3(G)$ denote the cardinality of a largest subset of an abelian group G containing no three-term arithmetic progression. Is there a constant $c < 3$ such that $r_3((\mathbb{Z}/3\mathbb{Z})^n)$ grows in n no faster than c^n?

General statement

Let $\alpha, \beta, \gamma \in \mathbb{F}_q$ such that $\alpha + \beta + \gamma = 0$ and $\gamma \neq 0$. Let A be a largest subset of $\mathbb{F}_q^n$ such that the equation $\alpha a_1 + \beta a_2 + \gamma a_3 = 0$ has no solutions with $a_1, a_2, a_3 \in A$ apart from those with $a_1 = a_2 = a_3$. Is there a constant $c < q$ such that $|A|$ grows in n no faster than c^n?

The cap set problem

Specific statement

Let $r_3(G)$ denote the cardinality of a largest subset of an abelian group G containing no three-term arithmetic progression. Is there a constant $c < 3$ such that $r_3((\mathbb{Z}/3\mathbb{Z})^n)$ grows in n no faster than c^n?

General statement

Let $\alpha, \beta, \gamma \in \mathbb{F}_q$ such that $\alpha + \beta + \gamma = 0$ and $\gamma \neq 0$. Let A be a largest subset of $\mathbb{F}_q^n$ such that the equation $\alpha a_1 + \beta a_2 + \gamma a_3 = 0$ has no solutions with $a_1, a_2, a_3 \in A$ apart from those with $a_1 = a_2 = a_3$. Is there a constant $c < q$ such that $|A|$ grows in n no faster than c^n?

Theorem (Ellenberg and Gijswijt, *Annals of Mathematics*, 2017)

Yes.

The cap set problem

Ellenberg and Gijswijt follow a breakthrough due to Croot, Lev, and Pach.

Idea: translate the problem to one about systems or spaces of polynomials. (the *polynomial method*)

1. Bound the size of the cap set by the dimension of a subspace of polynomials with coefficients in $\mathbb{F}_q$.
2. Control the asymptotic behavior of this bound.

The cap set problem

1. Bound the size of the cap set by the dimension of a subspace of polynomials with coefficients in $\mathbb{F}_q$.
2. Control the asymptotic behavior of this bound.

Ellenberg and Gijswijt use only “elementary” methods in step 1.

Tao, Zeilberger, and others have proposed elementary methods for step 2.

We further elementarize step 2.

The cap set problem in Lean

```
theorem general_cap_set {α : Type} [discrete_field α] [fintype α] :
∃ C B : ℝ, B > 0 ∧ C > 0 ∧ C < fintype.card α ∧
  ∀ {a b c : α} {n : ℕ} {A : finset (fin n → α)},
    c ≠ 0 → a + b + c = 0 →
    (∀ x y z : fin n → α, x ∈ A → y ∈ A → z ∈ A →
                  a · x + b · y + c · z = 0 → x = y ∧ x = z) →
    ↑A.card ≤ B * C^n
```

Formalization: constructing the bound

Preliminaries

Goal:

```
theorem theorem_12_1 {α : Type} [discrete_field α] [fintype α]
  (n : ℕ) {a b c : α} (hc : c ≠ 0) (habc : a + b + c = 0)
  (hn : n > 0) {A : finset (fin n → α)}
  (ha : ∀ x y z ∈ A, a · x + b · y + c · z = 0 → x = y ∧ x = z) :
  A.card ≤ 3 * m α n (1 / 3 * ((card α - 1) * n))
```

We fix a parameter `α : Type` instantiating the type classes `[discrete_field α]` and `[fintype α]`, and `n : ℕ`. We use `q : ℕ` to abbreviate `card α`.

Preliminaries

For `d : ℚ`, we make the following definitions:

- `M` is the set of monomials in `n` variables where the exponent of each variable is less than `q`.
- `M'` is the subset of `M` whose elements have total degree at most `d`.
- `S'` is the span of `M'`. This is a subspace of `mv_polynomial (fin n)` α.
- `m` is the dimension of `S'`.

Since `M'` is linearly independent, it follows that the cardinality of `M'` is equal to `m`.

Preliminaries

```
def M : finset (mv_polynomial (fin n) α) :=
(finset.univ.image
  (λ f : fin n →₀ fin q, f.map_range fin.val rfl)).image
    (λ v : fin n →₀ ℕ, monomial v (1:α))

def M' (d : ℚ) : finset (mv_polynomial (fin n) α) :=
M.filter (λ m, d ≥ mv_polynomial.total_degree m)

def S' (d : ℚ) : subspace α (mv_polynomial (fin n) α) :=
submodule.span α ((M' d) : set (mv_polynomial (fin n) α))

def m (d : ℚ) : ℕ := (vector_space.dim α (S' d)).to_nat

lemma M'_card (d : ℚ) : (M' d).card = m d
```

Preliminaries

```
parameters (T : subspace α (mv_polynomial (fin n) α))
           (A : finset (fin n → α))

def zero_set : set (mv_polynomial (fin n) α) :=
{p ∈ T.carrier | ∀ a ∈ A, mv_polynomial.eval a p = 0}

def zero_set_subspace : subspace α (mv_polynomial (fin n) α) :=
{ carrier := zero_set,
  zero := ⟨submodule.zero, by simp⟩,
  add := λ _ _ hx hy,
   ⟨submodule.add hx.1 hy.1, λ _ hp, by simp [hx.2 hp, hy.2 hp]⟩,
  smul := λ _ _ hp,
   ⟨submodule.smul hp.1, λ _ hx, by simp [hp.2 hx]⟩ }
```

Preliminaries

Our goal was:

```
theorem theorem_12_1 {α : Type} [discrete_field α] [fintype α]
  (n : ℕ) {a b c : α} (hc : c ≠ 0) (habc : a + b + c = 0)
  (hn : n > 0) {A : finset (fin n → α)}
  (ha : ∀ x y z ∈ A, a · x + b · y + c · z = 0 → x = y ∧ x = z) :
  A.card ≤ 3 * m α n (1 / 3 * ((card α - 1) * n))
```

Fix the hypotheses, and define:

```
def neg_cA : finset (fin n → α) := A.image (λ z, (-c) · z)
```

```
def V : subspace α (S' d) :=
zero_set_subspace (S' d) (finset.univ \ neg_cA)
```

```
def V_dim : ℕ := (vector_space.dim α V).to_nat
```

We prove a sequence of lemmas controlling `V_dim`.

Bounding from below

A general theorem (following from rank-nullity):

```
theorem lemma_9_2 (T : subspace α (mv_polynomial (fin n) α))
  (A : finset (fin n → α)) :
  (vector_space.dim α zero_set_subspace).to_nat + A.card ≥
    (vector_space.dim α T).to_nat
```

From this, we derive:

```
lemma diff_card_comp : (finset.univ \ neg_cA).card + A.card = q^n :=
by rw [finset.card_univ_diff, fintype.card_fin_arrow, neg_cA_card,
  nat.sub_add_cancel A_card_le_α_card_n]; refl

theorem lemma_12_2 :  q^n + V_dim ≥ m d + A.card :=
have V_dim + (finset.univ \ neg_cA).card ≥ m d,
  from lemma_9_2 _ _ V_dim_finite,
by linarith [diff_card_comp]
```

Bounding from above

There is a polynomial in `V` with maximal support:

```
lemma exi_max_sup :
  ∃ P ∈ V, ∀ P' ∈ V, sup P ⊆ sup P' → sup P = sup P'
```

Define `P` to be a witness to this.

```
theorem lemma_12_3 : (sup P).card ≥ V_dim
```

Bounding from above

```
theorem lemma_12_4 : (sup P).card ≤ 2 * m (d/2)
```

This follows from a more general result:

```
theorem prop_11_1 {p : mv_polynomial (fin n) α} (A : finset (fin n → α)) :
  p ∈ S' n d → (∀ x ∈ A, ∀ y ∈ A, x ≠ y → p.eval (a · x + b · y) = 0) →
  (A.filter (λ x, p.eval (-c · x) ≠ 0)).card ≤ 2 * m (d / 2)
```

Proposition (Ellenberg and Gijswijt)

Let $A \subseteq \mathbb{F}_q^n$ and $\alpha, \beta, \gamma \in \mathbb{F}_q$ with $\alpha + \beta + \gamma = 0$. Let $P \in S_n^d$ such that for all $a, b \in A$ with $a \neq b$ we have $P(\alpha a + \beta b) = 0$. Then

$$|\{a \in A \mid P(-\gamma a) \neq 0\}| \leq 2m_{d/2}.$$

Proposition 11.1

- This was the most intricate proof in our development.
  - (In line with E-G. This lemma makes up most of their paper.)
- Stated in terms of the linear transformation `p.eval`, but more naturally proved with matrices.
- Needed to extend libraries to unify these two concepts.

Proposition 11.1 proof sketch

Given `a b : α`, `x y : fin n → α`, `p : mv_polynomial (fin n) α` with `p ∈ S' d`:

- `p.eval (a · x + b · y)` can be written as a linear combination of evaluated monomials in `M' d`.
- Define an `A × A` matrix `B` such that `B x y = p.eval (a · x + b · y)`.
- Prove that `B` factors:

```
lemma B_eq_sum_matrix : B =
  split_left.sum (λ _ _, matrix.vec_mul_vec _ _) +
  split_right.sum (λ  _ _, matrix.vec_mul_vec _ _)
```

- Cardinalities of the finite sets `split_left` and `split_right` are at most `m (d/2)`.
- Rank of `B` is at most `2 * m (d/2)`, since `matrix.vec_mul_vec` has rank at most 1.
- But `B` is diagonal, so its rank is equal to what we want to bound.

A combinatorial calculation

The last lemma relates values of `m` at different inputs.

```
theorem lemma_12_5 : q^n ≤ m ((q-1)*n - d) + m d
```

- Largely independent of the previous lemmas.
- Go by carving up the space `fin n → fin q` into subsets.
- The encoding matters!

Putting things together

```
theorem lemma_12_6 : A.card ≤ 2 * m (d/2) + m ((q-1)*n - d) :=
by linarith using [lemma_12_2, lemma_12_3, lemma_12_4, lemma_12_5]
```

Abstracting the parameter `d` and instantiating it with `2/3*(q-1)*n`:

```
theorem theorem_12_1 : A.card ≤ 3*(m (1/3*((q-1)*n)))
```

Intermission: how do the proofs look?

Formalization: asymptotics

Controlling the growth of our bound

We want to know how our bound grows in `n`.

```
theorem theorem_12_1 : A.card ≤ 3*(m (1/3*((q-1)*n)))
```

Recall:

- `m d` is the number of monomials with total degree at most `d`.
- `q` is the cardinality of the underlying field α.

Controlling the growth of our bound

We want to know how our bound grows in `n`.

```
theorem theorem_12_1 : A.card ≤ 3*(m n (1/3*((q-1)*n)))
```

Recall:

- `m n d` is the number of monomials in `n` variables with total degree at most `d`.
- `q` is the cardinality of the underlying field α.

Controlling the growth of our bound

```
theorem general_cap_set {α : Type} [discrete_field α] [fintype α] :
  ∃ B C : ℝ, B > 0 ∧ C > 0 ∧ C < card α ∧
    ∀ {a b c : α} {n : ℕ} {A : finset (fin n → α)},
      c ≠ 0 → a + b + c = 0 →
      (∀ x y z ∈ A, a • x + b • y + c • z = 0 → x = y ∧ x = z) →
      A.card ≤ B * C^n
```

It suffices:

```
theorem general_cap_set' {α : Type} [discrete_field α] [fintype α] :
  ∃ B C : ℝ, B > 0 ∧ C > 0 ∧ C < card α ∧ 3*(m n (1/3*((q-1)*n))) ≤ B * C^n
```

m as a sum of coefficients

We will rewrite `m` as a sum of coefficients of a certain polynomial.

Informally, we define:

$$c_j^{(n)} := \left| \left\{ (a_1, \ldots, a_n) \;\middle|\; a_i \in \{0, 1, \ldots, q-1\} \text{ and } \sum_{i=1}^{n} a_i = j \right\} \right| .$$

How to encode these tuples in Lean?

m as a sum of coefficients

```
def sf (n j : ℕ) : finset (vector (fin q) n) :=
finset.univ.filter (λ f, (f.nat_sum = j))

def cf (n j : ℕ) : ℕ := (sf n j).card
```

where `vector A n` is defined as a subtype of lists:

```
def vector (α : Type u) (n : ℕ) := { l : list α // l.length = n }

def vector.cons : α → vector α n → vector α (nat.succ n)
| a ⟨ v, h ⟩ := ⟨ a::v, congr_arg nat.succ h ⟩
```

m as a sum of coefficients

```
theorem lemma_13_8 (n : ℕ) {d : ℚ} (hd : d ≥ 0) :
  m n d = (finset.range (⌊d⌋.nat_abs + 1)).sum (cf n)
```

The proof applies a result from before:

```
lemma h_B_card : m n d = (univ : finset (fin n → fin q)).filter (λ v,
    (total_degree (monom v)) ≤ d)
```

We establish an isomorphism between the two vector representations.

m as a sum of coefficients

```
def sf (n j : ℕ) : finset (vector (fin q) n) :=
finset.univ.filter (λ f, (f.nat_sum = j))

def cf (n j : ℕ) : ℕ := (sf n j).card

lemma cf_mul (n j : ℕ) : cf (n+2) j =
  (finset.range (j + 1)).sum (λ i, (cf 1 (j - i)) * cf (n + 1) i)
```

This involves lifting `n`-tuples to `n+1`-tuples. Much easier to do with the vector representation.

m as a sum of coefficients

We relate `cf n j` to coefficients of the polynomial $\left(1 + x + \ldots + x^{q-1}\right)^n$:

```
def one_coeff_poly (m : ℕ) : polynomial ℕ :=
(finset.range m).sum (λ k, (polynomial.X : polynomial ℕ) ^ k)

theorem lemma_13_9 (hq : q > 0) (n j : ℕ) :
  ((one_coeff_poly q) ^ n).coeff j = cf n j
```

m as a sum of coefficients

```
theorem lemma_13_10 (n : ℕ) {r : ℝ} (hr : r > 0) :
  cf n j ≤ (((one_coeff_poly q)^n).eval₂ coe r) / r^j
```

Obtained via a detour into complex numbers:

```
def ζk (k : ℤ) : ℂ := exp (2*π*I/k)

lemma pick_out_coef {f : polynomial ℂ} {i k : ℕ} (h1 : k > i)
  (h2 : k > nat_degree f) {r : ℝ} (h3 : r > 0) :
  (coeff f i) * k =
    (range k).sum (λ j, (eval (r*(ζk k)^j) f)/(r^i * ((ζk k)^(i*j)))
```

(and some tedious inequality computations)

Concrete bounds on m

Defining

```
def crq (r : ℝ) (q : ℕ) :=
((one_coeff_poly q).eval₂ coe r) / r ^ ((q-1)/3)
```

and combining

- `cf n j ≤ (((one_coeff_poly q)^n).eval₂ coe r) / r^j`
- `m n d = (finset.range (⌊d⌋.nat_abs + 1)).sum (cf n)`

we get

```
theorem theorem_13_13 (n : ℕ) {r : ℝ} (hr : 0 < r) (hr2 : r < 1) :
  (m n ((q - 1)*n / 3)) ≤ ((crq r q)^2 / (1 - r)) * (crq r q)^n
```

Concrete bounds on m

Since `crq 1 q = q` and the derivative of `crq` with respect to `r` is positive at `r = 1`, we have from elementary calculus:

```
theorem lemma_13_15 : ∃ r : ℝ, 0 < r ∧ r < 1 ∧ crq r q < q
```

Along with the previous theorem and `theorem_12_1`, we have proved our desired result:

```
theorem theorem_13_13 (n : ℕ) {r : ℝ} (hr : 0 < r) (hr2 : r < 1) :
  (m n ((q - 1)*n / 3)) ≤ ((crq r q)^2 / (1 - r)) * (crq r q)^n
```

```
theorem theorem_12_1 : A.card ≤ 3*(m n (1/3*((q-1)*n)))
```

Even more concrete bounds

For the motivating case when `q = 3`, we compute the optimal value `r := (real.sqrt 33 - 1) / 8`.

We show `0 < r < 1` and `crq r 3 = ((3 / 8)^3 * (207 + 33*real.sqrt 33))^(1/3)` (which is approximately 2.76).

```
theorem cap_set {n : ℕ} {A : finset (fin n → ℤ/3ℤ)} :
    (∀ x y z ∈ A, x + y + z = 0 → x = y ∧ x = z) →
     A.card ≤ 198 * (((3/8) ^ 3 * (207 + 33 * sqrt 33)) ^ (1/3)) ^ n
```

Morals

Statistics

- Ellenberg–Gijswijt proof: about 2 pages of content. (construction of bound: 1.5 pages)
- Our informal writeup: 10 pages of non-background content (construction of bound: 5 pages)
- Our formalization: 2500 lines (construction of bound: 900 lines)

Morals

- This is formalized contemporary math—rare!
- It was “smooth” (for a formalization).
- As is often the case: library development may have been the biggest gain.
- Collaboration was essential.