Formalisation of the Bannai-Bannai-Stanton Theorem in Lean 4

Antoine du Fresne von Hohenesche

January 2, 2026

Abstract

This document outlines my design decisions and formalisation strategy for the proof of the Bannai–Bannai–Stanton theorem on s-distance sets in $\mathbb{R}^d$. The formalisation follows the algebraic proof by Petrov and Pohoata, comprising approximately 1000 lines of (non-optimised) Lean 4 code. I also discuss the workflow, including the utilisation of Large Language Models in the formalisation process.

1 Introduction

The project formalises the upper bound for the cardinality of an s-distance set $S \subset \mathbb{R}^d$. Specifically, we prove that if $d \in \mathbb{N}$ and S is any subset of $\mathbb{R}^d$ such that the set of distances

$$D(S) := \{r \in \mathbb{R} \mid \exists x \in S, \exists y \in S, x \neq y \wedge r = \|x - y\|_{\mathbb{R}^d}\}$$

(where $\|\cdot\|_{\mathbb{R}^d}$ is the standard Euclidean norm on $\mathbb{R}^d$) is finite, then

$$|S| \leq \binom{d + |D(S)|}{|D(S)|} = \binom{d + |D(S)|}{d}.$$

2 Structure of the Formalisation

Instead of the original proof, we follow the modern algebraic approach by Petrov and Pohoata [1], which utilises polynomial spaces and rank bounds on specific matrices. The proof strategy involves establishing general linear-algebraic lemmas and specialising them to specific "distance" polynomials.

2.1 Identification of Missing Components

Organising the project was essentially a reverse-engineering process of the main goal. By analysing the logical dependencies of the proof, I identified four key components required for the formalisation of the Bannai–Bannai–Stanton theorem. I subsequently structured the code into five distinct sections to address these needs, the last one being the main statement.

Finiteness of S (The Diagonal Argument): The algebraic proof by Petrov and Pohoata assumes that S is finite in order to construct a certain matrix in $\mathbb{R}^{S \times S}$. However, the general theorem can be applied to any $S \subset \mathbb{R}^d$ with a finite distance set. A prerequisite step was therefore to prove that $|D(S)| < \infty \implies |S| < \infty$. I formalised a "Diagonal Argument" lemma and then established the goal. I could have restricted myself to finite sets, but I wanted to be as general as possible.

Monomial Counting: A formal lemma was required to count multivariate monomials of a bounded total degree (the "Stars and Bars" argument). Since this counting was already handled

in another form for multisets in Mathlib's `Sym.card_sym_eq_choose`, I decided to count the number of such monomials by constructing a bijection with multisets. This component involved a significant number of equivalences between `Finsupp` and `Multiset`. I utilised Gemini Pro 3 (in GitHub Copilot in VS Code) heavily here to bridge the vocabulary gap between the intuitive combinatorial argument and Mathlib's technical implementation. I could have tried to prove the lemma from scratch by induction, but I did not want to complicate matters, and I honestly did not evaluate which of the two approaches would be easier: the induction or the bijection strategy.

Generalised Croot–Lev–Pach Lemma: While an apparently specific instance of this lemma exists in the formalisation of the Cap Set problem, it is likely that the version used by Petrov and Pohoata is more general, as they note in their paper. In any case, I could not find such an instance, and I decided to prove it from scratch for two reasons: completeness and further generalisation. I was able to extend one of the two results of this lemma to hold over any totally ordered field (rather than just $\mathbb{R}$).

```
lemma Croot_Lev_Pach_lemma_generalized_1st_part ... :
  M.rank ≤ 2 * dim A s := ...
lemma Croot_Lev_Pach_lemma_generalized_2nd_part ... :
  (∀ (V : Submodule F (A → F)), (∀ v  V, v ≠ 0 → Q v > 0) →
    Module.finrank F V ≤ dim A s)
  (∀ (V : Submodule F (A → F)), (∀ v  V, v ≠ 0 → Q v < 0) →
    Module.finrank F V ≤ dim A s)
```

This relied on four preliminary lemmas which I will not detail here. Three of them could technically be added to Mathlib (with some optimisation): bounding the rank of the sum of two matrices by the sum of their ranks, the standard dimension decomposition for non-degenerate symmetric bilinear forms, and a technical lemma splitting the evaluation of a monomial in $2d$ variables. The final preliminary lemma was long and specific to the combinatorial setting and the bilinear form used in the proof.

Polynomial Definitions: To apply the general algebraic framework to our specific geometric problem, I defined "distance polynomials" representing the squared Euclidean distance. This involved two main constructs: a polynomial in $2d$ variables (`distPoly`) representing the distance between two variable points, and a specialised family of polynomials (`distPoly_from`) representing the distance from a fixed point. I utilised Mathlib's `MvPolynomial` library to establish bounds on their total degrees and to prove evaluation lemmas, namely that these polynomials compute the corresponding squared Euclidean distances, and that evaluating the $2d$-variable polynomial at (x, y) is equivalent to evaluating the fixed-point polynomial associated with y at x.

The Main Result: The final component is the integration of these tools. It combines the finiteness lemma (allowing S to be cast to a `Finset`), the polynomial definitions, and the Croot–Lev–Pach bounds with the dimension bound to derive the final inequality.

```
theorem bannai_bannai_stanton_bound {d s : ℕ}
  (S : Set (EuclideanSpace ℝ (Fin d)))
  (hdis : (Set.distances S).Finite)
  (hs : s = (hdis.toFinset).card) :
  S.encard ≤ Nat.choose (d + s) s := ...
```

3 Workflow and LLM Usage

As requested, here is an overview of how LLMs were integrated into the development process.

3.1 What Worked

- **Mathlib Lemma Discovery:** My workflow often involved writing the proof strategy in natural language and asking the AI to help translate it into Lean or to suggest tactics. Throughout this process, I also used search tools (mostly Loogle, Google, ChatGPT, DeepSeek) (e.g. I described a mathematical fact (e.g. "dimension of the dual space") and the LLM suggested relevant Mathlib names), including AI-assisted grep searches (in the terminal) via Gemini in GitHub Copilot (I use VS Code), to analyse existing Mathlib lemmas.
- The LLM was particularly helpful (almost did all of the work) in constructing the bijection for the monomial-counting argument (using `Option`).
- For the rank bound (Croot–Lev–Pach Lemma, Part 1), I first attempted to prove the first part of the Croot–Lev–Pach lemma myself using the argument from the paper. However, the AI suggested a different and "shorter" approach (at least at the formalization level compared to what I was writing) to bounding the rank (see comments in the code): instead of using the bilinear form, it directly decomposes the matrix as a sum of two matrices, carefully isolating the relevant parts of the polynomial. I do not know whether this is the original idea of Croot, Lev, and Pach, or whether this argument is standard in the literature but the mathematical insight for this specific step came from the model. As said this approach was significantly easier to formalise, since I had already developed much of the necessary machinery (in particular, evaluation lemmas for these polynomials, degree bounds, and dimension bounds for the associated vector spaces of functions).

3.2 What Did Not Work

- **Hallucination:** The LLM often failed to solve small intermediate goals in a single attempt, as it frequently referred to non-existent Mathlib lemmas (for example, incorrect name for the rank of the sum of matrix bounds). This required to verify a lot of the suggestions in the documentation.
- **Logical Flow:** The LLMs tended to produce very long proofs by reducing goals to extremely simple statements and tackling them step by step and it often failed at doing so as well. A human mathematician would likely employ higher-level reasoning and apply more powerful abstract tools. As a result, the generated proofs were often unnecessarily lengthy.

4 Conclusion and Future Work

The proof is complete and compiles without `sorry`. I recently discussed upstreaming this result with the Mathlib community (specifically Kevin Buzzard). While the proof is correct, the style is currently "unidiomatic" for the central library. Future work would involve refactoring the proof to match Mathlib's strict style guidelines, and clean up the code.

References

[1] Fedor Petrov and Cosmin Pohoata. A remark on sets with few distances in $\mathbb{R}^d$. *Proceedings of the American Mathematical Society*, 149(2):569–572, 2021. doi: 10.1090/proc/15234. URL https://arxiv.org/abs/1912.08181.