# Formalising Szemerédi's Regularity Lemma in Lean

**Yaël Dillies**
Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, UK

**Bhavik Mehta**
Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, UK

## Abstract

Szemerédi's Regularity Lemma is a fundamental result in graph theory with extensive applications to combinatorics and number theory. In essence, it says that all graphs can be approximated by well-behaved unions of random bipartite graphs. We present a formalisation in the Lean theorem prover of a strong version of this lemma in which each part of the union must be approximately the same size. This stronger version has not been formalised previously in any theorem prover. Our proof closely follows the pen-and-paper method, allowing our formalisation to provide an explicit upper bound on the number of parts. An application of this lemma is also formalised, namely Roth's theorem on arithmetic progressions in qualitative form via the triangle removal lemma.

**2012 ACM Subject Classification** Theory of computation → Interactive proof systems; Mathematics of computing → Extremal graph theory

**Keywords and phrases** Lean, formalisation, formal proof, graph theory, combinatorics, additive combinatorics, Szemerédi's Regularity Lemma, Roth's Theorem

**Digital Object Identifier** 10.4230/LIPIcs.ITP.2022.9

**Supplementary Material** A snapshot of the development branch containing all the formalisation relevant here is available at:
*Software (Source Code)*: `https://github.com/b-mehta/regularity-lemma`
archived at `swh:1:dir:e1da55f4222dac91b940ca052928eaace09762da`

**Funding** *Bhavik Mehta*: Cantab Capital Institute for the Mathematics of Information

**Acknowledgements** We want to thank Anne Baanen, Thomas Bloom, Kevin Buzzard, Timothy Gowers, Heather Macbeth, the mathlib maintainers and the anonymous reviewers for useful comments on previous versions of this paper. We are grateful to the mathlib community, whose work has been invaluable for the prerequisites of this project.

## 1 Introduction

Extremal Combinatorics is a modern and rapidly developing area of discrete mathematics [2], with problems that are often motivated by questions in other areas, such as Geometry, Number Theory, Theoretical Computer Science and Game Theory. It studies the maximal (or minimal) size of a collection of objects (such as natural numbers, edges in a graph or subsets of a finite set), subject to particular restrictions. An especially simple question in this area is Mantel's theorem [21]: What is the maximum number of edges in a graph $G$ on $n$ vertices which does not contain a triangle (i.e. three mutually adjacent vertices $a, b, c$)? Another well-studied problem is given by Roth's theorem on arithmetic progressions, which asks for the maximum size of a subset $A \subseteq \{1, \dots, n\}$ which does not contain three distinct integers $a, b, c$ such that $a + c = 2b$.

While Mantel's theorem was resolved in 1907 – the maximum number of edges is given by $\lfloor x^2/4 \rfloor$ – we do not have precise bounds for Roth's theorem, but rather asymptotic lower and upper bounds with a nontrivial gap between them [16, 3]. Nonetheless, some of the earliest work on this problem showed that the maximum size must be smaller than any linear function of $n$ – in fact in 1953 Roth proved it is bounded above by a constant multiple of


$\frac{n}{\log\log n}$ [27]. One proof of Roth's theorem uses Szemerédi's regularity lemma [29], one of the most powerful tools of extremal graph theory [20]. A simplified statement of this lemma says that every graph can be partitioned into parts that behave similarly in many ways to random graphs, which are often easier to prove results about than arbitrary graphs.

In this paper we describe our formal proof in the Lean theorem prover [23] of a strong form of the regularity lemma, and provide the application of the lemma to Roth's theorem. We use results from the Lean mathematical library mathlib [22], which is characterised by a distributed and decentralised community of contributors, and ubiquitous classical reasoning. Some of the more general results formalised have already been accepted into mathlib, and the remainder are in the process of being merged. The up-to-date development branch is publicly available on GitHub[1], but we also provide a snapshot of our code online[2].

A simultaneous and independent formalisation of a version of Szemerédi's regularity lemma and Roth's theorem in Isabelle/HOL has been completed by Chelsea Edmonds, Angeliki Koutsoukou-Argyraki and Lawrence Paulson, also at the University of Cambridge [10, 9]. While the statements of Roth's theorem which have been formalised are equivalent, the version of the regularity lemma which we have formalised produces a partition with stronger properties: namely the parts of the partition are all almost exactly the same size. This extra property comes at the cost of a larger bound, but both bounds are so large that this is not mathematically significant.

Our primary source for this formalisation was lectures given by Andrew Thomason on Extremal Graph Theory at the University of Cambridge in 2019, especially for the proof of the regularity lemma which we were able to follow with only minor changes. We also used notes by Yufei Zhao [32], particularly for the triangle removal lemma and corners theorem.

Throughout the paper we provide code blocks of Lean from our formalisation, with some adjustments for the sake of readability. In particular, proofs are omitted using `sorry` when only the type is relevant, implicit arguments are omitted when the types are clear from the context, and occasionally declarations are given shorter names.

In Section 2 we go into the mathematical details of Szemerédi's regularity lemma, the triangle removal lemma and Roth's theorem, giving mathematical statements of each and indicating the key proof ideas which we have formalised. Section 3 describes the formalised version of the regularity lemma and we give its statement in Lean and explain some of the details of the formalisation. Similarly, Section 4 discusses the triangle counting and triangle removal lemmas, the latter of which is one of the primary applications of the regularity lemma, while Section 5 describes the application away from graph theory to the corners theorem and Roth's theorem. In Section 6, we go into more detail about how our development compares with the Isabelle formalisation, discuss related and future work, and summarise our findings.

## 2 Mathematical details

Let us now give the results which are vital to this paper. We assume familiarity with basic graph theory; a detailed account of this field may be found in [4]. We take all graphs to be finite and simple, we write $V(G)$ for the vertex set of the graph $G$, and we write $|G|$ as an abbreviation for $|V(G)|$, the number of vertices in the graph.

[1] `https://github.com/leanprover-community/mathlib/tree/szemeredi/src/combinatorics/szemeredi`
[2] `https://github.com/b-mehta/regularity-lemma`

An *equipartition* of a finite set $V$ of $n$ elements is a partition $V_1, \dots, V_k$ such that for each $i$, $\lfloor n/k \rfloor \le |V_i| \le \lceil n/k \rceil$. Equivalently, for each $i, j$ we have $||V_i| - |V_j|| \le 1$. We also refer to such partitions as *equitable*.

We now take vertex sets $X, Y \subseteq V(G)$. The *edge count* $e_G(X, Y)$ of the pair $(X, Y)$ is the number of edges (in $G$) between vertices in $X$ and vertices in $Y$, and the *density* of the pair is given by

$$d_G(X, Y) = \frac{e_G(X, Y)}{|X||Y|}.$$

Note that $0 \le d_G(X, Y) \le 1$. We will often omit the subscript $G$ in both of these if the context is clear.

For a real $\varepsilon > 0$, the pair of vertex sets $X, Y \subseteq V(G)$ is said to be *$\varepsilon$-uniform* if for every $A \subseteq X$ and $B \subseteq Y$ with $|A| \ge \varepsilon|X|$ and $|B| \ge \varepsilon|Y|$ we have

$$|d(A, B) - d(X, Y)| < \varepsilon.$$

The partition $V_1, \dots, V_k$ is then said to be *$\varepsilon$-uniform* (also called *$\varepsilon$-regular*) if no more than $\varepsilon\binom{k}{2}$ of the pairs $(V_i, V_j)$ for $i < j$ fail to be $\varepsilon$-uniform. In practice, we will typically have $\varepsilon$ very small (certainly $\ll 1$), and $n$ very large.

Intuitively, a pair of vertex sets is uniform if taking subsets does not affect the edge density significantly, provided we do not take a subset which is too small. This is a property satisfied by taking the obvious bipartition on a random bipartite graph, for instance, and it can be seen as a quasirandom or pseudo-random property [30].

## 2.1 Proof idea of the regularity lemma

▶ **Theorem 1** (Szemerédi [29]). *Let $\varepsilon > 0$, and let $l$ be a natural number. Then there exists an integer $L$ such that every graph $G$ with $|G| \ge l$ has an $\varepsilon$-uniform equipartition into $m$ parts for some $m$ such that $l \le m \le L$.*

Most importantly, note the choice of $L$ depends only on $\varepsilon$ and $l$ and must not depend on our choice of graph $G$ and in particular not on its size. In fact, our proof gives an explicit value of $L$ as a function of $\varepsilon$ and $l$ given as an exponential tower of height $\approx 4\varepsilon^{-5}$ with $l$ at the top of the tower, which we have also formalised, and it is known that any upper bounds on the regularity lemma must be of tower type [13].

In addition, note that the condition that $m \ge l$ is included in the statement in order to ensure that the number of parts is not too small. This condition is itself helpful to prevent the partition being too trivial: for instance the partition into exactly one part is always equitable and uniform.

The proof of the lemma uses a quantity often called the *energy* or *index* of a partition $P = \{V_1, \dots, V_k\}$ (with respect to the graph $G$) given by

$$q_G(P) = \frac{1}{k^2} \sum_{1 \le i < j \le k} d_G(V_i, V_j)^2.$$

Observe that the energy is nonnegative, and no larger than $\frac{1}{2}$.

The idea of the proof is not too difficult to describe now. We begin with a trivial equipartition $P_0$ into exactly $l$ parts, taking the parts arbitrarily so long as the partition is equitable. At stage $P_i$, either the partition is $\varepsilon$-uniform and we stop, or we will construct a refined equipartition $P_{i+1}$ with not many more parts and such that the energy of $P_{i+1}$ is at

least $\varepsilon^5/8$ more than the energy of $P_i$. This process must terminate, as the energy is bounded above by a constant, at which point the final partition has precisely the desired properties. In particular, the number of steps taken is no more than $4/\varepsilon^5$, and hence provided that we have a bound on the number of parts of $P_{i+1}$ in terms of $P_i$, we can deduce a bound on the number of parts in the final partition, which is independent of $G$.

All that remains to complete the proof sketch is to describe the refinement process. We do not give a detailed description of this process as it is fairly technical, and more detailed accounts can be found in [4], but instead give a flavour of the ideas used in the process. Given the equipartition $P_i$ which we know not to be $\varepsilon$-uniform with respect to $G$ we will give a new partition $Q$, which is a refinement of $P_i$, and has the desired energy increment (in fact, we will have an increment of $\varepsilon^5/2$, better than we had hoped for). To do this, for the non-uniform pairs we pick subsets that witness the non-uniformity, and then pick $Q$ such that these subsets are all unions of parts of $Q$. However, this partition $Q$ may not be equitable, so we must do an additional "shuffling" step to produce $P_{i+1}$ by producing an equipartition which closely approximates $Q$ in the appropriate sense. Of course, our $P_{i+1}$ may no longer have the required energy increment, but by very careful choice of the shuffling operation we may show that the energy of $P_{i+1}$ is close to the energy of $Q$, recovering the desired energy increment of $\varepsilon^5/8$.

## 2.2 Triangle removal

A well-known application of the regularity lemma is the triangle removal lemma. Recall that a *triangle* in a graph is given by three distinct vertices which are adjacent, and a graph is *triangle-free* if it has no triangles.

▶ **Theorem 2** (Triangle removal theorem[32]). *Let $0 < \varepsilon \leq 1$. Then there exists a $\delta > 0$ such that for any graph $G$ with $n$ vertices with fewer than $\delta n^3$ triangles, we can remove no more than $\varepsilon n^2$ edges in such a way that the resulting graph will be triangle-free.*

Intuitively, this says that if the proportion of triangles is small, then to remove all of them we only need to remove a small proportion of the edges (as the number of "possible" triangles in an arbitrary graph is on the order of $n^3$, and the number of possible edges is on the order of $n^2$). We now sketch the proof of this theorem using the regularity lemma to illustrate the sorts of ideas which need to be formalised. As before, we do not give all the details of the proof here, but they can be found in [32].

To prove this lemma, we suppose we have such a graph $G$, assuming for the sake of contradiction that there is no collection of fewer than $\varepsilon n^2$ edges which can be removed to make the graph triangle-free. As is common in this sort of combinatorial proof, we will choose $\delta$ later. We first use the regularity lemma to construct a partition which is $\varepsilon/8$-uniform and has at least $l = 4/\varepsilon$ parts. We then construct the "reduced graph" $G'$ of $G$ by removing all edges which are internal to parts of the partition, removing edges between non-uniform pairs, and removing edges between pairs with edge density less than $\varepsilon/4$. By the properties of the partition constructed by the regularity lemma, it is not hard to show that this step does not remove more than $\varepsilon n^2$ edges: the first set of edge removals is internal to a part, and we know that each part has size at most $\lceil n/l \rceil \approx \varepsilon n/4$; the second set operates only on non-uniform pairs of which there must be few; and the third set of removals by definition removes edges between pairs of small density so cannot remove many edges. From our contradiction hypothesis, the reduced graph must have a triangle, and by construction the triangle has each of its vertices in different parts of the partition.

We then appeal to the triangle counting lemma.

▶ **Lemma 3** (Triangle counting lemma). *Let $0 \le \varepsilon \le \frac{1}{2}$. Take a tripartite graph $G$ with vertex sets $X, Y, Z$ which are pairwise $\varepsilon$-uniform, and suppose the pairwise edge densities between them are not below $2\varepsilon$. Then there must be at least $(1-2\varepsilon)\varepsilon^3|X||Y||Z|$ triangles in $G$.*

This lemma can be proved by a fairly straightforward counting argument, and then applied to our situation (with the obvious change in $\varepsilon$) to deduce that there must be a cubic number of triangles, as the three parts that our given triangle were in satisfy the hypotheses of the triangle counting lemma, thus concluding the proof.

## 2.3 Roth's theorem

For convenience, we restate Roth's theorem in an qualitative form.

▶ **Theorem 4** (Roth [27]). *For every $\delta > 0$, there exists an $n_0$ such that for any $n \ge n_0$ and any subset $A \subseteq \{1, \dots, n\}$ satisfying $|A| \ge \delta n$, there are distinct elements $a, b, c \in A$ such that $a + c = 2b$.*

Note that an equivalent conclusion is that there are $a, d$ with $d \neq 0$ and $a, a+d, a+2d \in A$ – both of these conditions ask for three equally spaced elements of $A$, i.e. an arithmetic progression of length 3.

While it is possible to prove Roth's theorem directly from the triangle removal lemma [32], an alternative approach is to first prove the corners theorem [28]. The corners theorem itself is a compelling result and can be viewed as a generalisation of Roth's theorem, so to illustrate the power of the results we have formalised, we take this alternative path. For a subset $B \subseteq \{1, \dots, n\}^2$, a *corner* is a triple of points of the form $(x, y),\ (x+h, y),\ (x, y+h) \in B$ with $h > 0$.

▶ **Theorem 5** (Corners theorem [1]). *For every $\delta > 0$, there exists an $n_0$ such that for any $n \ge n_0$ and any subset $B \subseteq \{1, \dots, n\}$ satisfying $|B| \ge \delta n^2$, there is a corner in $B$.*

As previously, we sketch the flavour of the proof here rather than giving all details. Let us define an *anticorner* to be a triple of points of the form $(x, y), (x+h, y), (x, y+h) \in B$ with $h < 0$. To prove the corners theorem, it is convenient to first show the existence of *either* a corner or an anticorner in a sufficiently dense set, and then deduce the corners theorem by a straightforward mirroring argument. If $B \subseteq \{1, \dots, n\}$ has no corners, by the pigeonhole principle there exists a point $z \in \{1, \dots, 2n\}^2$ such that $B \cap (z - B)$ has size $\ge |B|^2/(2n)^2$, and this intersection contains no corners or anticorners.

Thus it suffices to show that our subset $B \subseteq \{1, \dots, n\}$ has either a corner or an anticorner – we will call this the weak corners theorem. To this end, we construct a tripartite graph $X, Y, Z$ such that triangles are in bijective correspondence with corners of $B$, anticorners of $B$ or "trivial" corners, namely those with $h = 0$. By the size condition on $B$, there must be at least $\delta n^2$ of these trivial corners and hence at least $\delta n^2$ triangles. Furthermore, these triangles are all edge-disjoint, so to remove all triangles we must remove at least $\delta n^2$ edges of the graph, and now the triangle removal lemma shows that we have $\varepsilon n^3$ triangles, so we have at least $\varepsilon n^3 - \delta n^2$ non-trivial triangles. Hence if $\varepsilon n > \delta$ there is a non-trivial corner or anti-corner, as required (and we choose $n_0$ such that $n_0 > \varepsilon/\delta$).

Finally, from the corners theorem we can prove Roth's theorem: Given our set $A \subseteq \{1, \dots, n\}$ construct the set $B = \{(x, y) : x - y \in A\} \subseteq \{1, \dots, 2n\}^2$. It is easy to see that corners in the set $B$ produce 3-term arithmetic progressions in $A$, and we can directly calculate that $|B| \ge n|A|$ to finish the proof.

In the following sections, we describe the Lean formalisation of these concepts and proofs.

# 3 Szemerédi's regularity lemma

We begin by describing the basic data types used for the formalisation of the regularity lemma, focusing on partitions; then discuss a technicality regarding choice of witnesses. Next we discuss in detail how the induction construction works, and the calculations involved in showing it has the required properties, and conclude this section by showing how these parts fit together to prove the main theorem.

As mentioned previously, we will omit some proofs by using the Lean keyword `sorry`, some arguments are omitted when the types are clear from the context, and occasionally declarations are given shorter names.

## 3.1 Partitions

The notion in mathlib of `finset` provides a type of finite sets which can used for finitary operations, such as a sum or a finite union, and considered as (coerced to) sets using the syntax `(s : set α)`, a *type ascription*. There already exist partition-like concepts in mathlib, namely `setoid` and `indexed_partition`, but neither of those satisfactorily support finitely many parts. As the regularity lemma is inherently finite, we settled on defining a new kind of partition.

A finite partition of $s$ is a finite set of pairwise disjoint finite sets (called parts) whose (finite) union is $s$. For technical reasons and following usual mathematical convention, we forbid the trivial part. A partition is equitable if all its parts have almost the same size, in our case written by saying the function `card` is equitable on the set of parts.

Our parts in a partition will not be indexed by $\{1, \dots, k\}$, but rather simply considered as a collection. This is a design decision which we found most convenient to work with, for instance so we do not worry about indexing when constructing new partitions from old.

```
structure finpartition (s : finset α) :=
(parts : finset (finset α))
(disjoint : parts.pairwise_disjoint)
(sup_parts : parts.sup id = s)
(not_bot_mem : ⊥ ∉ parts)

def set.equitable_on (s : set α) (f : α → ℕ) : Prop :=
∀ a₁ a₂, a₁ ∈ s → a₂ ∈ s → f a₁ ≤ f a₂ + 1

def finpartition.is_equipartition
  {s : finset α} (P : finpartition s) : Prop :=
(P.parts : set (finset α)).equitable_on card
```

It is useful at this point to briefly discuss Lean's "dot notation". Given a term `x` of type `my_type` say, along with a definition `my_type.func` taking `x` as an explicit argument, we may abbreviate `my_type.func x` as `x.func` for convenience's sake. In this case, we will write `P.is_equipartition` because `P` is a `finpartition`.

We now define the energy as a real number. To simplify later algebraic manipulations, the expression we use will in fact be double the mathematical definition given previously, we sum over $U \neq V$ thanks to `finset.off_diag`: given a finite set, this will construct the finite set of ordered pairs which are unequal. Using the indexed partition notation, this corresponds more closely to $\sum_{1 \leq i \neq j \leq k}$.

```
def finset.off_diag (s : finset α) : finset (α × α) := sorry
```

As a consequence, note that each refinement will increase the energy by $\frac{\varepsilon^5}{4}$, not $\frac{\varepsilon^5}{8}$, and the energy is bounded above by 1, rather than $\frac{1}{2}$.

```
def finpartition.energy (P : finpartition s) (G : simple_graph α) : ℝ :=
(∑ UV in P.parts.off_diag, (G.edge_density UV.1 UV.2)^2) / P.parts.card^2
```

## 3.2 Witnesses

When the pair $(U, V)$ is non-uniform, the proof reads "Pick $U' \subseteq U, V' \subseteq V$ witnesses of non-uniformity". If one does that naïvely, extracting $(U', V')$ from the existential in $(U, V)$, the witnesses we get from $(U, V)$ and $(V, U)$ will not in general match. Instead of getting a pair of witnesses we get two halves of one, which will not be good enough to prove the desired properties about the choices. So we must pick the witnesses in such a way as to make the witnesses picked from both sides agree.

Our first try was using `sym2`: `sym2 X` is the type of unordered pairs of `X`. This seems promising to avoid our problem with ordered pairs, as we can make the ordered witnesses for the ordered pair $(U, V)$ out of unordered witnesses for the unordered pair $(U, V)$. But now we must match the witnesses to the original pairs which, while possible to implement, is awkward to work with in practice.

In the end, we went for a much simpler solution: Put an arbitrary order on the parts and only take the witnesses of $(U, V)$ if $U < V$; otherwise take the witnesses of $(V, U)$ and swap them.

The lemma `not_witness_prop` simply converts a proof that the pair $(U, V)$ is not uniform to an existential statement asserting that witnesses to non-uniformity exist, so we can use the axiom of choice to pick them.

```
def witness_aux (ε : ℝ) (U V : finset α) : finset α × finset α :=
if h : ¬G.is_uniform ε U V
  then ((not_witness_prop h).some, (not_witness_prop h).some_spec.2.some)
  else (U, V)

def witness (ε : ℝ) (U V : finset α) : finset α :=
if well_ordering_rel U V
  then (G.witness_aux ε U V).1
  else (G.witness_aux ε V U).2
```

From mathlib we have `well_ordering_rel` which constructs a well ordering on any given type using the axiom of choice, and `well_ordering_rel U V` is then a predicate for whether $U < V$ in this ordering.

## 3.3 Constructing the new partition

The induction step is rather delicate, we need to construct the new partition $Q$ from the existing partition $P$ very carefully to have all of the appropriate requirements: ensuring it is equitable is the most difficult.

We proceed in five steps, where the first four will operate inside a fixed part $U$ of $P$, and the fifth combines all of these to make the final partition $Q$.

1. First, we collect the witnesses of non-uniformity with respect to $U$. For each part $V$ of $P$ such that $(U, V)$ is not $\varepsilon$-uniform, we take the corresponding witness.

```
def finpartition.witnesses (P : finpartition (univ : finset α))
  (G : simple_graph α) (ε : ℝ) (U : finset α) : finset (finset α) :=
(P.parts.filter (λ V, U ≠ V ∧ ¬G.is_uniform ε U V)).image (G.witness ε
    U)
```

2. Second, from a set of sets $F$, we can construct the partition of a set $s$ such that its points lie in the same part if and only they are in the same sets of $F$. In particular, this means that every set in $F$ will be a union of parts from the partition. We construct the partition by going through every $E \subseteq F$ (written as `F.powerset`, and constructing the set `s.filter (λ a, ∀ t ∈ F, t ∈ E ↔ a ∈ t)` of the elements of `s` which are in every element of $E$ and no element of $F \setminus E$, to produce a part. We must also remove $\emptyset$, which is done by the `of_erase` constructor, and we elide the proofs that this is indeed a partition.

```
def atomise (s : finset α) (F : finset (finset α)) : finpartition s :=
of_erase
    (F.powerset.image (λ E, s.filter (λ a, ∀ t ∈ F, t ∈ E ↔ a ∈ t)))
    sorry
```

3. For a partition $P$ of a set $s$ of size $am + b(m + 1)$, find a partition $Q$ of $s$ into $a$ parts of size $m$, $b$ parts of size $m + 1$ such that each part of $P$ is the union of the parts of $Q$ it contains along with at most $m$ "spare" vertices.

```
def equitabilise (s : finset α) (P : finpartition s)
  (m a b : ℕ) (h : a * m + b * (m + 1) = s.card) :
  finpartition s := sorry
```

   The construction is fairly easy in informal mathematics, but not immediate to formalise: an additional complication was that in our reference the precise condition on $Q$ was not explicitly given, and the size condition "h" was not spelled out particularly clearly – although both of these were implicit in the argument and did not require any change.
   The construction we used was also not described in our reference as it is intuitive – particularly at this level of mathematics – but we needed to give the construction ourselves and formalise it. The idea is to repeatedly remove those sets of size $m$ or $m + 1$ from $s$ which are entirely contained in one part of $P$, as these can immediately form a part of $Q$. This terminates (as the size of the remainder strictly decreases), and from what is left we may form parts of $Q$ arbitrarily: the size condition on $s$ ensures that we have a good number of vertices left, enough to form entire parts.
   To formalise this idea, we use strong induction on the (size of the) finite set `s`, removing a set of size $m$ if $a > 0$, and a set of size $m + 1$ otherwise, and hence construct the desired partition and give its required properties.
4. Putting all this together, we get a refinement of each part $U$ of the partition. The refinement is obtained by taking the witnesses of non-uniformity, atomising using these, and equitabilising the resulting partition. Depending on whether $U$ was a "big" or a "small" part, the precise parameters for the equitabilise step change, which is the content of the outer `if ... then ... else ...`.

```
def finpartition.chunk_increment (P : finpartition (univ : finset α))
  (hP : P.is_equipartition) (G : simple_graph α) (ε : ℝ)
  (U : finset α) (hU : U ∈ P.parts) :
  finpartition U :=
if U.card = card α / P.parts.card
```

```
  then (atomise U (P.witnesses G ε U)).equitabilise m
         (4^P.parts.card - a) a sorry
  else (atomise U (P.witnesses G ε U)).equitabilise m
         (4^P.parts.card - a - 1) (a + 1) sorry
```

Note, the `hP` and `hU` hypotheses are used in the elided `sorry` to satisfy the cardinality condition from `equitabilise`.

5. Finally, juxtaposing the refinement of each part yields the desired partition refinement. We do so using `finpartition.bind`, which takes a partition $P$ and partitions of its parts as input, and returns the partition obtained by juxtaposing those subpartitions.

```
def finpartition.increment (P : finpartition (univ : finset α)) (hP :
    P.is_equipartition) (G : simple_graph α) (ε : R) : finpartition
    (univ : finset α) :=
P.bind (P.chunk_increment hP G ε)
```

## 3.4 Calculations

The goal is to prove that one refinement step increases the energy by at least $\frac{\varepsilon^5}{4}$. To do this, we break the energy into two sums: one over uniform parts and one over non-uniform ones. We bound each sum term-wise.

```
variables (P : finpartition univ) (hP : P.is_equipartition)
  (hPG : ¬P.is_uniform G ε)

lemma energy_increment_uniform
  {U V : finset α} (hU : U ∈ P.parts) (hV : V ∈ P.parts) :
  (G.edge_density U V)^2 - ε^5/25 ≤
    (∑ (a, b) in (P.chunk_increment hP G ε hU).parts ×
    (P.chunk_increment hP G ε hV).parts,
      (G.edge_density a b)^2) / 16^P.parts.card := sorry

lemma energy_increment_nonuniform
  {U V : finset α} (hU : U ∈ P.parts) (hV : V ∈ P.parts)
  (hUV : U ≠ V) (hGUV : ¬ G.is_uniform ε U V) :
  (G.edge_density U V)^2 - ε^5/25 + ε^4/3 ≤
    (∑ (a, b) in (P.chunk_increment hP G ε hU).parts ×
    (P.chunk_increment hP G ε hV).parts,
      (G.edge_density a b)^2) / 16^P.parts.card := sorry

lemma energy_increment :
  P.energy G + ε^5 / 4 ≤ (P.increment G ε).energy G := sorry
```

A great deal of the proof is spent on lower bounding the energy of the refined partition. This part was the most tedious and the least mathematically interesting, both on paper and during formalization. It accounts for roughly 700 lines of code. Such length is mostly explained by the equitability requirement: the non-equitable regularity lemma requires much less fiddly calculations. Nevertheless we can partly attribute this to our unfamiliarity with formal manipulations of complicated sums at the start of this project, and we found mathlib automation not particularly suited to our case. For instance, tactics for linear arithmetic such as `linarith` are not currently useful to deal with sums, while rewrite lemmas as commonly used by the simplifier are difficult to use to show chains of inequalities by transitivity.

## 3.5 Induction

Our end theorem is *effective*, in the sense that we give an explicit bound for $n_0$, rather than merely stating its existence. To be able to take a refined partition at each stage, we have a technical condition that $100 < 4^n\varepsilon^5$ and we also want the end partition to have at least $l$ parts, so we start the induction with a dummy partition whose size is expressed by `initial_bound`. Recall that a refinement step on a partition in $n$ parts yields a partition in $n4^n$ parts. This is expressed by `exp_bound` below. As each refinement step increases the energy by $\frac{\varepsilon^5}{4}$ and the energy stays between 0 and 1, we won't need more than $\frac{4}{\varepsilon^5}$ refinements, which makes `szemeredi_bound` our final bound.

```
def exp_bound (n : ℕ) : ℕ := n * 4^n

def initial_bound (ε : ℝ) (l : ℕ) : ℕ :=
max l (⌊log (100/ε^5) / log 4⌋₊ + 1)

def szemeredi_bound (ε : ℝ) (l : ℕ) : ℕ :=
let L := exp_bound^[⌊4 / ε^5⌋₊] (iteration_bound ε l) in L * 16^L
```

Here, `⌊x⌋₊` is notation for the floor function. and `f^[n]` is notation for function iteration, namely the composition $\underbrace{f \circ \cdots \circ f}_{n \text{ times}}$.

We are now in a position to state and prove the regularity lemma.

```
theorem szemeredi_regularity [fintype α] (G : simple_graph α)
  (ε : ℝ) (hε : 0 < ε) (l : ℕ) (hl : l ≤ card α) :
  ∃ (P : finpartition univ),
    P.is_equipartition ∧ l ≤ P.parts.card ∧ P.parts.card ≤
    szemeredi_bound ε l ∧ P.is_uniform G ε :=
```

This is then mathematically proved by iteration on energy. To lead the process, we formally use an induction over the naturals, where our specific goal hypothesis is displayed here:

```
∀ i : ℕ, ∃ (P : finpartition univ), P.is_equipartition ∧
  t ≤ P.parts.card ∧ P.parts.card ≤ exp_bound^[i] t ∧
    (P.is_uniform G ε ∨ ε^5 / 4 * i ≤ P.energy G)
```

The $i = 0$ case can be proved by taking an arbitrarily chosen equipartition with $t$ parts which clearly satisfies all the requirements, while the inductive case will be satisfied by using `finpartition.increment`, together with `energy_increment` or using the original partition. Then, applying this with `i` chosen to be `⌊4 / ε^5⌋₊ + 1`, the right hand side of the $\vee$ must fail as the energy is bounded above by 1, so we must have a uniform partition.

# 4 Triangle counting and triangle removal

In this section we describe the formalisation of the triangle counting and triangle removal lemmas.

We begin by defining $n$-cliques in a graph, an obvious generalisation of triangles, by giving a predicate describing when a given finite set of vertices forms an $n$-clique: there are $n$ of them and they are pairwise adjacent. It is then easy to define a predicate for a graph having no triangles: any finite set of vertices cannot form a 3-clique.

```
def simple_graph.is_n_clique (G : simple_graph α) (n : ℕ) (s : finset α) :
  Prop :=
s.card = n ∧ (s : set α).pairwise G.adj

def simple_graph.no_triangles (G : simple_graph α) : Prop :=
∀ t, ¬ G.is_n_clique 3 t
```

We use the coercion of `finset` to `set` as mentioned previously, as the `pairwise` predicate is defined for arbitrary sets. While this generic definition is not directly useful to this application, it will no doubt be useful to other graph theorists working with mathlib: for instance to state Ramsey's theorem or Turán's theorem [31]. Recall also that we may write `G.is_n_clique 3 t` since `G` is a simple graph using Lean's dot notation.

The (finite) set of triangles in a graph can then be defined, and we can prove the key property of this set, that a triple `{x, y, z}` is in the set of triangles if and only if the three obvious adjacency relations hold.

```
def simple_graph.triangle_finset (G : simple_graph α) :
  finset (finset α) := sorry

lemma mem_triangle_finset (x y z : α) :
  {x, y, z} ∈ G.triangle_finset ↔ G.adj x y ∧ G.adj x z ∧ G.adj y z :=
sorry
```

Following the logical progression, we now state and prove the triangle counting lemma. While the mathematical statement is usually given in terms of a tripartite graph, in applications (including ours) one often wants to take the three vertex sets as disjoint subsets of a single graph, so an alternative statement is helpful as well.

```
lemma triangle_counting
  {X Y Z : finset α} {ε : ℝ} (hε₀ : 0 < ε) (hε₁ : ε ≤ 1)
  (dXY : 2 * ε ≤ G.edge_density X Y) (uXY : G.is_uniform ε X Y)
  (dXZ : 2 * ε ≤ G.edge_density X Z) (uXZ : G.is_uniform ε X Z)
  (dYZ : 2 * ε ≤ G.edge_density Y Z) (uYZ : G.is_uniform ε Y Z) :
  (1 - 2 * ε) * ε^3 * X.card * Y.card * Z.card ≤
    ((X.product (Y.product Z)).filter $
    λ (x, y, z),
        G.adj x y ∧ G.adj x z ∧ G.adj y z).card :=
sorry

lemma triangle_counting2 {X Y Z : finset α} {ε : ℝ}
  (dXY : 2 * ε ≤ G.edge_density X Y) (uXY : G.is_uniform ε X Y)
  (hXY : disjoint X Y)
  (dXZ : 2 * ε ≤ G.edge_density X Z) (uXZ : G.is_uniform ε X Z)
  (hXZ : disjoint X Z)
  (dYZ : 2 * ε ≤ G.edge_density Y Z) (uYZ : G.is_uniform ε Y Z)
  (hYZ : disjoint Y Z) :
  (1 - 2 * ε) * ε^3 * X.card * Y.card * Z.card ≤ G.triangle_finset.card :=
sorry
```

In the statement of `triangle_counting`, we have explicitly worked with the edge relation of `G`, as opposed to re-using the previous definition of triangles: this is simply to emphasise the point that the "triangles" considered in this lemma are in fact *ordered* triples, rather than merely finite sets of size 3. In contrast, in `triangle_counting2`, we do use `G.triangle_finset.card` as we are now counting triangles of the graph `G`.

Next we must define the *reduced graph* as discussed in Section 2.

```
def reduced_graph (ε : ℝ) (P : finpartition (univ : finset α)) :
    simple_graph α :=
{ adj := λ x y, G.adj x y ∧
    ∃ U V ∈ P.parts, x ∈ U ∧ y ∈ V ∧ U ≠ V ∧
        G.is_uniform (ε/8) U V ∧
        ε/4 ≤ G.edge_density U V,
  symm := sorry,
  loopless := sorry }

lemma reduced_graph_le {ε : ℝ} {P : finpartition (univ : finset α)} :
  reduced_graph G ε P ≤ G :=
λ x y ⟨h, _⟩, h
```

To define a simple graph we give the adjacency relation, and prove it is symmetric and loopless: these proofs are omitted as they are completely routine. To express the adjacency relation, we say that `x` and `y` are adjacent in the reduced graph exactly when they are adjacent in the original graph, and that there exist parts `U, V` of the partition `P` containing `x, y` respectively, which are distinct, uniform and have an `edge_density` which is at least $\varepsilon/4$. This captures precisely the definition as given on paper, as the chosen parts must be unique by definition of a partition, and the distinctness property ensures that we are removing any edge both of whose endpoints are in a single part. We may then easily prove that the reduced graph is a subgraph of the original graph, given as a $\leq$ relation.

The other key property of the reduced graph we need is that the number of removed edges is less than $\varepsilon n^2$.

```
lemma reduced_edges_card [nonempty α]
  {ε : ℝ} {P : finpartition (univ : finset α)}
  (hε : 0 < ε) (hP : P.is_equipartition) (hPε : P.is_uniform G (ε/8))
  (hP' : 4 / ε ≤ P.parts.card) :
  (G.edge_finset.card - (reduced_graph G ε P).edge_finset.card : ℝ) <
    ε * (card α)^2 :=
```

As is to be expected, `G.edge_finset` refers to the finite set of edges of the graph `G`. The explicit type ascription on the left of the inequality is needed for Lean to infer that we are discussing an inequality of reals, otherwise it will assume that we are stating an inequality of naturals as the left-hand-side is a natural number.

At this point we are able to prove the triangle removal lemma, but we instead pause to give a reformulation which is more convenient to prove (both mathematically and formally): the two versions are easily seen to be equivalent, indeed they are simply contrapositives of one another.

Given $\varepsilon > 0$, we say a graph $G$ is *$\varepsilon$-far from being triangle-free* if one must delete at least $\varepsilon|G|^2$ edges of $G$ to remove all triangles.

▶ **Theorem 6** (triangle removal, restated). *Let $0 < \varepsilon \leq 1$. Then there exists a $\delta > 0$ such that any graph $G$ which is $\varepsilon$-far from being triangle-free contains at least $\delta|G|^3$ triangles.*

To express the property of being far from triangle-free, we say that any subgraph $G'$ of $G$ which is triangle-free must have no more than $\varepsilon|G|^2$ edges fewer than $G$.

```
def triangle_free_far (G : simple_graph α) (ε : ℝ) : Prop :=
∀ (G' ≤ G), G'.no_triangles →
    ε * (card α)^2 ≤ (G.edge_finset.card - G'.edge_finset.card : ℝ)
```

While the statement alone only says that some $\delta$ must exist for the given $\varepsilon$, the proof in fact gives an explicit form for what $\delta$ must be. We thus find it convenient to express $\delta$ as a function of $\varepsilon$, but have made no attempt to optimise this bound, but simply observe it has the same asymptotic behaviour (in the big-O sense) as proofs which follow the same approach in the standard literature. However a 2010 result of Fox [12] proves (a generalisation) of the triangle removal lemma which does not use the regularity lemma, and hence is able to achieve a significant improvement on the bound on $\delta$. Thus, the precise value of the bound given should not be taken too seriously, but we provide it for completeness.

```
def triangle_removal_bound (ε : ℝ) : ℝ :=
min (1 / (2 * ⌈4/ε⌉₊^3))
    ((1 - ε/4) * (ε/(16 * szemeredi_bound (ε/8) ⌈4/ε⌉₊))^3)
```

Finally we have all the tools available to state and prove both forms of the lemma of this section.

```
lemma triangle_removal_2 {ε : ℝ} (hε : 0 < ε) (hε₁ : ε ≤ 1)
  (hG : G.triangle_free_far ε) :
  triangle_removal_bound ε * (card α)^3 ≤ G.triangle_finset.card :=
sorry

lemma triangle_removal {ε : ℝ} (hε : 0 < ε) (hε₁ : ε ≤ 1)
  (hG : (G.triangle_finset.card : ℝ) <
           triangle_removal_bound ε * (card α)^3) :
  ∃ (G' ≤ G),
    (G.edge_finset.card - G'.edge_finset.card : ℝ) < ε * (card α)^2
      ∧ G'.no_triangles :=
sorry
```

With these theorems formalised, all that remains is to deduce the corners theorem and Roth's theorem.

## 5 Corners theorem and Roth's theorem

In this section we describe the formal proof of the corners theorem and Roth's theorem. As indicated in Section 2, the proof of the corners theorem works by first showing a "weak" corners theorem in which an anticorner may be constructed by means of constructing an appropriate graph and applying the triangle removal lemma.

For a finite set $A$ of pairs of naturals, we define when a triple forms a corner or anticorner.

```
def is_corner (A : finset (ℕ × ℕ)) : ℕ → ℕ → ℕ → Prop :=
λ x y h, (x, y) ∈ A ∧ (x + h,y) ∈ A ∧ (x, y + h) ∈ A

def is_anticorner (A : finset (ℕ × ℕ)) : ℕ → ℕ → ℕ → Prop :=
λ x y h, (x, y) ∈ A ∧ (h ≤ x ∧ (x-h,y) ∈ A) ∧ (h ≤ y ∧ (x, y-h) ∈ A)
```

It is important to note here that the formalised definition is different from the mathematical one given previously. In particular, we allow `h` to be zero in both cases, and so in the formal statements of the theorems we must take care to disallow this case, else the theorems become trivially true. Nonetheless, it is useful to allow this case in the definition, as "trivial" corners with $h = 0$ are important to talk about in the proof in order to show the graph we will construct is far from being triangle free.

The reader may also be surprised to see the condition $h \leq x$ appearing in the definition of anticorners. This is present as Lean's subtraction operation on naturals is truncated subtraction, also known as the monus operator, and gives $x - y = 0$ in the case $x \leq y$ so the mathematical objection of "$A$ is a set of pairs of naturals, so cannot contain negatives" does not apply. Thus we must insist that $h \leq x$, otherwise the anticorner may behave unpredictably.

The auxiliary tripartite graph we construct from the set $A$ to prove the weak corners theorem has the first vertex set as horizontal lines of $\{1, \ldots, n\}^2$, the second vertex set as vertical lines of the same set, and the third vertex set as diagonal lines. This latter vertex set will be indexed by the integers $\{1, \ldots, 2n\}$, and the diagonal line corresponding to vertex $k$ is represented by the set of points $(x, y)$ with $x + y = k$. Edges of the graph are then given by pairs of lines in different directions which intersect at a point of $A$.

We can construct this in Lean by using its built-in inductive type mechanism, which defines a new type by specifying its constructors. The definition `corners_edge` below in fact has six constructors, we will show just two of them, the others are analogous.

```
inductive corners_vertices (N : ℕ)
| horiz : fin N → corners_vertices
| vert : fin N → corners_vertices
| diag : fin (2 * N) → corners_vertices

inductive corners_edge (A : finset (ℕ × ℕ)) :
  corners_vertices N → corners_vertices N → Prop
| hv {h v : fin N} : (h, v) ∈ A →
       corners_edge (horiz h) (vert v)
| hd {h : fin N} {k : fin (2 * N)} : h ≤ k → (h, k - h) ∈ A →
       corners_edge (horiz h) (diag k)

def corners_graph (N : ℕ) (A : finset (ℕ × ℕ)) :
  simple_graph (corners_vertices N) :=
{ adj := corners_edge A, symm := sorry, loopless := sorry }
```

The type `fin N` is the type of naturals strictly less than $N$, and corresponds to our notion of $\{1, \ldots, n\}$. We have omitted some type ascriptions here for the sake of readability, but emphasise that all inequalities and subtractions here take place over the natural numbers, and there is of course a canonical embedding from `fin N` to the naturals. As previously, we insist on inequalities in naturals as additional hypotheses in order to avoid issues with natural subtraction.

To give an example, the part of the `corners_edge` definition labelled `hd` (horizontal-diagonal) allows construction of an edge between the vertex represented by the horizontal line `h` and the vertex represented by the diagonal line `k` exactly when `h` $\leq$ `k` and `(h, k - h)` $\in$ `A`, i.e. exactly when the two lines intersect at a point in `A`. The other five cases cover the remaining line intersections. Thus the Lean definition mimics the mathematical definition, the sole exception being zero-indexing in Lean rather than one-indexing in the mathematics.

As discussed in the proof sketch, we now show that the `corners_graph` is far from being triangle-free, in particular $\varepsilon/16$-far. It is useful first to show that if we can find $t$-many edge-disjoint triangles, and $e|G|^2 \leq t$, then our graph is $\varepsilon$-far from being triangle free, and then deduce that if $\varepsilon n_0^2 \leq |A|$, the corners graph of $A$ is $\varepsilon/16$-far from being triangle free. The former proof can be done simply by observing that if we have removed edges and removed all the triangles, we must have removed at least one edge from every triangle (by edge-disjointness), and as we have many triangles we immediately show the number of removed edges must satisfy the required inequality.

```
lemma triangle_free_far_of_disjoint_triangles {ε : ℝ}
  (tris : finset (finset α)) (htris : tris ⊆ G.triangle_finset)
  (pd : (tris : set (finset α)).pairwise (λ x y, (x ∩ y).card ≤ 1))
  (tris_big : ε * card α ^ 2 ≤ tris.card) :
  G.triangle_free_far ε := sorry

lemma disjoint_triangles {ε : ℝ} (hA : A ⊆ (range N).product (range N))
    (hA' : ε * N^2 ≤ A.card) :
  (corners_graph N A).triangle_free_far (ε/16) := sorry
```

With this result at hand, it is now easy to show the weak corners theorem, as the triangle removal lemma allows us to conclude that there are non-trivial triangles. We show that if every corner or anticorner is trivial, then the number of triangles is bounded above by $n^2$, thus conclude that if `ε * n^2 ≤ A.card` with sufficiently large $n$ we are done.

```
lemma trivial_triangles_corners_graph
  {A : finset (ℕ × ℕ)} {n : ℕ}
  (cs : ∀ (x y h : ℕ), is_corner A x y h → h = 0)
  (as : ∀ (x y h : ℕ), is_anticorner A x y h → h = 0) :
  (corners_graph n A).triangle_finset.card ≤ n^2 := sorry

lemma weak_corners_theorem {ε : ℝ} (hε : 0 < ε) :
  ∃ n₀ : ℕ, ∀ n, n₀ ≤ n →
    ∀ A ⊆ (range n).product (range n), ε * n^2 ≤ A.card →
      ∃ x y h, 0 < h ∧ (is_corner A x y h ∨ is_anticorner A x y h) :=
```

Now, the proof of the corners theorem and Roth's theorem can be done, to conclude our formal proofs.

```
lemma corners_theorem {ε : ℝ} (hε : 0 < ε) :
  ∃ n₀ : ℕ, ∀ n, n₀ ≤ n →
    ∀ A ⊆ (range n).product (range n), ε * n^2 ≤ A.card →
      ∃ x y h, 0 < h ∧ is_corner A x y h :=
sorry

lemma roth (δ : ℝ) (hδ : 0 < δ) :
  ∃ n₀ : ℕ, ∀ n, n₀ ≤ n →
    ∀ A ⊆ range n, δ * n ≤ A.card →
      ∃ a d, 0 < d ∧ a ∈ A ∧ a + d ∈ A ∧ a + 2 * d ∈ A :=
sorry
```

## 6 Discussion

We conclude by discussing related work; the advantages of mathlib for work like ours, suggesting future work and concluding.

### 6.1 Comparison with Isabelle

Edmonds, Koutsoukou-Argyraki, and Paulson independently and simultaneously formalised Szemerédi's regularity lemma and Roth's theorem in Isabelle/HOL [10, 9]. Due to the striking coincidence, this particular piece of related work is deserving of extra discussion.

While there are many similarities in our work, our approaches differ on several points:

- We formalised the equitable regularity lemma. This means that all parts in the final partition are ensured to be the same size, up to a difference of 1.
- We picked a different definition of uniformity of partitions.
- We showed the corners theorem on the way to Roth's theorem.

A particular deficiency in the current mathlib tactic system is for natural number arithmetic. For instance, one specific lemma that is needed for our formalization is

```
example (i j : ℕ) (hj : 0 < j) :
  j * (j - 1) * (i / j + 1) ^ 2 < (i + j) ^ 2 :=
sorry
```

which took 6 lines of proof, primarily working manually with elementary inequalities on real numbers, while strong automation could be able to simplify proofs such as these.

### 6.1.1 Equitability

There are a number of variants of the regularity lemma: the Isabelle/HOL development proves a version in which the resulting partition need not be equitable. From our "equitable" version, it is easy to deduce the non-equitable version, but not vice-versa: to the best of the authors' knowledge, the proof must be entirely re-done rather than strengthened to make this upgrade. We also note that mathematically, the extra restriction makes the induction step more difficult, both in its construction and proof: we must perform the `equitabilise` step, and work through a series of inequalities to recover the energy boost. Together, this represents more than half of the proof of the regularity lemma on paper, and empirically took the majority of person-hours to formalise. While part of this time may be attributed simply to the fact that the proof is relatively complex, we also believe it is in part due to both authors' unfamiliarity with real number calculations in Lean at the start of the project.

### 6.1.2 Uniformity

There are two standard definitions of uniformity of a partition:

**(a)** The first one says that there is at most a proportion of $\varepsilon$ pairs of non-uniform parts.

**(b)** The second one says that

$$\sum_{\substack{u,v\in P\\ \neg(u,v)\ \varepsilon\text{-uniform}}} |u||v| \leq \varepsilon |P|^2.$$

We chose the former while Edmonds et al. went for the latter. When the partitions in question are all equitable (as ours are), these two definitions are equivalent, up to a factor of 2 in the parameter $\varepsilon$ one way and 4 the other way. However when the partitions are not equitable, neither trivially implies the other: definition (a) will allow a small number of very large non-uniform pairs while definition (b) allows many very small non-uniform pairs.

## 6.2 Related work

The cap set problem, formalised in Lean in 2019 [7], has many mathematical similarities to Roth's theorem: they both ask about the maximum size of a subset of a particular set which avoids three-term arithmetic progressions. However, the method of proof in the two problems are very different: Roth's theorem uses graph theory and the regularity method, whereas the cap set problem uses linear algebra and the polynomial method.

Lean's graph theory library is relatively new, but has been used to show Hall's Marriage Theorem including infinitary versions [17], and the Friendship theorem[3]. Many other theorem provers have developed graph theory libraries, including PVS [5], Mizar [18], Isabelle [25], Coq [8]. As far as we are aware, there has not been any other formalisation progress in the area of extremal graph theory or extremal combinatorics.

## 6.3 Concluding thoughts

The choice of the regularity lemma was in part to provide another powerful tool in the Lean mathematician's toolbox, as well as to investigate how subtle induction arguments may be adapted to formalised proofs. An obvious future step is to formalise additional proofs which use the regularity lemma, such as the blow-up lemma [19] and the Erdős-Stone theorem [11], or applications outside of extremal graph theory such as bounds on the Ramsey number of graphs of bounded maximum degree [6].

More broadly, the ideas of this paper may be adapted to formalise generalisations of the results given here, such as hypergraph regularity lemmas of Gowers [14] or Nagle, Rödl, Schacht and Skokan [24, 26] in order to deduce Szemerédi's theorem on arithmetic progressions, which in turn begins the path towards formalising the celebrated Green-Tao theorem [15].

Our overall formalisation is about 3500 lines of code (including blank lines and comments), of which around 900 lines are for the triangle counting and removal theorems and 500 are for the corners theorem and Roth's theorem. We validated existing design choices used in mathlib primarily for finite sets and simple graphs and added minor lemmas which were not previously present. Parts of our work has already been accepted into mathlib, and we are in the process of merging the remainder.

We have additionally confirmed that combinatorics in Lean can often be done by following pen-and-paper proofs, or mathematical proofs in existing literature rather than needing to come up with new abstractions. Our work indicates the value of a single library of mathematics able to work with finite sets, real number inequalities and structures such as partitions and simple graphs to enable proofs using all of these in tandem.

## References


1 Miklós Ajtai and Endre Szemerédi. Sets of lattice points that form no squares. *Stud. Sci. Math. Hungar*, 9(1975):9–11, 1974.

2 Noga Alon. Problems and results in extremal combinatorics–III. *Journal of Combinatorics*, 7(2):233–256, 2016.

3 Thomas F Bloom and Olof Sisask. Breaking the logarithmic barrier in Roth's theorem on arithmetic progressions. *arXiv preprint arXiv:2007.03528*, 2020.

4 Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 1998.

5 Ricky W Butler and Jon A Sjogren. *A PVS graph theory library*, volume 206923. National Aeronautics and Space Administration, Langley Research Center, 1998.

6 C Chvatál, Vojtech Rödl, Endre Szemerédi, and W Tom Trotter Jr. The Ramsey number of a graph with bounded maximum degree. *Journal of Combinatorial Theory, Series B*, 34(3):239–243, 1983.


[3] https://github.com/leanprover-community/mathlib/blob/master/archive/100-theorems-list/83_friendship_graphs.lean

7 Sander R. Dahmen, Johannes Hölzl, and Robert Y. Lewis. Formalizing the Solution to the Cap Set Problem. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving (ITP 2019)*, volume 141 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITP.2019.15.

8 Christian Doczkal and Damien Pous. Graph theory in coq: Minors, treewidth, and isomorphisms. *Journal of Automated Reasoning*, 64(5):795–825, 2020.

9 Chelsea Edmonds, Angeliki Koutsoukou-Argyraki, and Lawrence C. Paulson. Roth's Theorem on Arithmetic Progressions. *Archive of Formal Proofs*, December 2021. , Formal proof development. URL: https://isa-afp.org/entries/Roth_Arithmetic_Progressions.html.

10 Chelsea Edmonds, Angeliki Koutsoukou-Argyraki, and Lawrence C. Paulson. Szemerédi's Regularity Lemma. *Archive of Formal Proofs*, November 2021. , Formal proof development. URL: https://isa-afp.org/entries/Szemeredi_Regularity.html.

11 Paul Erdös and Arthur H Stone. On the structure of linear graphs. *Bulletin of the American Mathematical Society*, 52(12):1087–1091, 1946.

12 Jacob Fox. A new proof of the graph removal lemma. *Annals of Mathematics*, pages 561–579, 2011.

13 William Timothy Gowers. Lower bounds of tower type for Szemerédi's uniformity lemma. *Geometric & Functional Analysis GAFA*, 7(2):322–337, 1997.

14 William Timothy Gowers. Hypergraph regularity and the multidimensional Szemerédi theorem. *Annals of Mathematics*, pages 897–946, 2007.

15 Ben Green and Terence Tao. The primes contain arbitrarily long arithmetic progressions. *Annals of mathematics*, pages 481–547, 2008.

16 Ben Green and Julia Wolf. A note on Elkin's improvement of Behrend's construction. In *Additive number theory*, pages 141–144. Springer, 2010.

17 Alena Gusakov, Bhavik Mehta, and Kyle A. Miller. Formalizing Hall's Marriage Theorem in Lean, 2021. arXiv:2101.00127.

18 Sebastian Koch. Unification of graphs and relations in mizar. *Formalized Mathematics*, 28(2):173–186, 2020.

19 János Komlós. The blow-up lemma. *Combinatorics, Probability and Computing*, 8(1-2):161–176, 1999.

20 János Komlós, Ali Shokoufandeh, Miklós Simonovits, and Endre Szemerédi. The regularity lemma and its applications in graph theory. In *Summer school on theoretical aspects of computer science*, pages 84–112. Springer, 2000.

21 W. Mantel. Problem 28 (Solution by H. Gouwentak, W. Mantel, J. Teixeira de Mattes, F. Schuh and W. A. Wythoff). *Wiskundige Opgaven*, 10:60–61, 1907.

22 The mathlib Community. The Lean Mathematical Library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pages 367–381, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3372885.3373824.

23 Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In *International Conference on Automated Deduction*, pages 378–388. Springer, 2015.

24 Brendan Nagle, Vojtěch Rödl, and Mathias Schacht. The counting lemma for regular k-uniform hypergraphs. *Random Structures & Algorithms*, 28(2):113–179, 2006.

25 Lars Noschinski. A graph library for isabelle. *Mathematics in Computer Science*, 9(1):23–39, 2015.

26 Vojtěch Rödl and Jozef Skokan. Regularity lemma for k-uniform hypergraphs. *Random Structures & Algorithms*, 25(1):1–42, 2004.

27 Klaus F Roth. On certain sets of integers. *J. London Math. Soc*, 28(104-109):3, 1953.

28 József Solymosi. Note on a generalization of Roth's theorem. In *Discrete and computational geometry*, pages 825–827. Springer, 2003.

**29** Endre Szemerédi. Regular partitions of graphs. Technical report, Stanford Univ Calif Dept of Computer Science, 1975.

**30** Andrew Thomason. Pseudo-random graphs. In *North-Holland Mathematics Studies*, volume 144, pages 307–331. Elsevier, 1987.

**31** Paul Turán. On an extremal problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941.

**32** Yufei Zhao. Graph Theory and Additive Combinatorics, 2019. URL: `https://yufeizhao.com/gtac/gtac.pdf`.