The seventh ternary density Hales–Jewett number

P. Graham
https://orcid.org/0009-0007-8493-1505

October 28, 2026

Abstract

Let $c_{n,3}$ be the largest size of a subset of $\{0,1,2\}^n$ containing no combinatorial line, a triple obtained by varying a nonempty set of coordinates through the same letter while fixing the rest. We give a computer-assisted proof that $c_{7,3} = 1302$, confirming Polymath1's tentative conjecture and closing its bounds $1302 \le c_{7,3} \le 1348$, recorded by Tao in 2009 and O'Bryant in 2015. The witness realizes Polymath1's known lower bound. Three auxiliary SAT bounds, slicing and global alphabet symmetry give sixteen main slice obligations. Refutations of all nineteen formulas were accepted by FRAT-rs, and in a separate earlier production campaign by drat-trim's `lrat-check`. Formal verification applies to certificate checking: cake_lpr, verified in HOL4 and compiled with CakeML, accepted refutations of all nineteen formulas across two regeneration campaigns. The SAT encodings, cap induction, symmetry reduction, case cover and counting interpretation are justified by the written proof and separate executable checks; they are outside that formal verification.

The proofs, the search code, the certificate checkers and the text were produced by AI models (from OpenAI and Anthropic), working under the author's direction. This work has not yet been reviewed by experts.

1 Introduction

The density Hales–Jewett theorem of Furstenberg and Katznelson [1], with a later combinatorial proof by Polymath1 [2], concerns large subsets of word spaces and the combinatorial lines they must contain. A combinatorial line varies a nonempty set of coordinates through the same letter while fixing the rest. The finite extremal values are not determined by the asymptotic theorem. For a fixed dimension n, write $c_{n,3}$ for the maximum size of a line-free subset of the ternary word cube $\{0,1,2\}^n$, and abbreviate $c_n = c_{n,3}$, with $c_0 = 1$. The Polymath1 project, writing as D.H.J. Polymath, had determined these numbers through dimension six and established the lower bound 1302 in dimension seven by March 2009 [4]; its 2010 paper gives a full account [3, Theorem 1.4 and Sections 2–3]. Section 2 of its 2010 paper calls this the bound "which we tentatively conjecture to be the correct bound" [3]. OEIS A156989, accessed October 20, 2026, lists the established terms through dimension six, ending at 450 [6].

The historical gap between Polymath1's bounds is explicitly documented. In a March 4, 2009 Polymath1 post, Tao writes: "The current best-known bounds for c_7 are $1302 \le c_7 \le 1348$." [4] O'Bryant's 2015 paper repeats "$1302 \le c_{7,3} \le 1348$" in its list of density Hales–Jewett numbers [5, Section 3.3, p. 7]. We close this gap by proving the matching unrestricted upper bound. The upper-bound proof was first posted in version 1, cited below. The Polymath1 paper gives the 1302 lower bound and the slicing inequality, but does not state 1348; the latter bound is recorded in Tao's contemporary account, without an individual discoverer identified here.

Relation to earlier versions. Version 1, posted October 20, 2026 [11], first presented the theorem and upper-bound proof. Version 2, also posted October 20 [12], corrected the attribution of the 2009 bounds and described the witness as a realization of Polymath1's known lower bound. Version 3, posted October 21 [13], completed cake_lpr coverage without changing the theorem or formulas; Section 5 distinguishes the checking campaigns. Version 4, posted October 22 [14], dated Polymath1's values through dimension six explicitly to March 2009, corrected the Shelah citation, and withdrew the earlier qualified first-proof claim. This version, version 5, corrects wording of credits, the review-status disclosure, descriptions of adjacent work, and the version and checker history. It clarifies the scope of formal verification and the proved-cap notation; the theorem, formulas and computational checks are unchanged.

Theorem 1. *Every subset of* $\{0,1,2\}^7$ *with no combinatorial line has at most 1302 points, and a line-free subset of size 1302 exists. Equivalently,* $c_{7,3} = 1302$.

The contribution claimed here is the upper bound for arbitrary subsets. The known lower bound and the conjectured value belong to Polymath1. The global alphabet permutation only orders slice labels without loss of generality; the set need not be invariant under it. The argument proves the caps $c_1 \le 2$, $c_4 \le 52$ and $c_5 \le 150$ by auxiliary SAT refutations, and obtains $c_2 \le 6$, $c_3 \le 18$ and $c_6 \le 450$ by slicing. The elementary base $c_1 \le 2$ is included as a SAT obligation for uniform certificate checking: the full three-point cube is itself a line. A set of size at least 1303 has largest first-coordinate slice size $s \ge \lceil 1303/3 \rceil = 435$, while the six-dimensional cap gives $s \le 450$. Ordering the three slice sizes therefore leaves sixteen cases. Each SAT instance imposes total size at least 1303 and exact largest-slice size $a_0 = s \ge a_1 \ge a_2$.

Separate production and regeneration campaigns provide the certificate checks summarized in Section 5. That section identifies each checker's coverage, the recorded CNF identity matches, and the measured costs.

The trusted base consists of the written cap induction, global symmetry and case-coverage arguments; the counting interpretation and its separately written, unverified audit program, which imports no encoder code; the proof checkers within the coverage stated; and the unverified exhaustive witness checkers. Each hypothetical set determines all auxiliary variables simultaneously by actual point counts. The auditor matches every clause symbolically to a clause family whose counting justification is written in Section 4; it does not sample sets. Its success supports the required extension from any set counterexample to a satisfying assignment, subject to those written lemmas and the correctness of the auditor. Separate implementation is a safeguard, not a proof of correctness or independent human review. The complete theorem, including this reduction, is not formalized. The small-dimensional caps are proved within the computation, rather than imported as unchecked premises from Polymath1.

Recent work addresses adjacent questions: Mouhib's symmetric and one-weight coloring bounds [7], Conlon–Farnsworth–Robertson's coloring questions [8], Shelah's improved bounds on the Hales–Jewett numbers [9], and Dahia's formalization of the density Hales–Jewett theorem [10]. These are different endpoints and are not premises of this proof. This result determines one finite extremal value, not a new proof or improved general bound for the density theorem.

2 Lines, caps and symmetry

Any line-free set of at least 1303 points has a line-free 1303-point subset. Our formulas directly exclude every set of size at least 1303, which is stronger than excluding sets of size exactly 1303.

A wildcard pattern is a word in $\{0,1,2,*\}^n$ with at least one wildcard. Its line consists of $w(0), w(1), w(2)$, replacing all wildcards simultaneously. There are $4^n - 3^n$ lines; thus the seven-dimensional audit checks 14,197 lines. A coordinate subcube is obtained by fixing some coordinates. A line-free set meets a d-dimensional subcube in at most $c_{d,3}$ points. Splitting on one coordinate gives

$$c_{d,3} \le 3c_{d-1,3}. \tag{1}$$

We continue to use the shorthand $c_d = c_{d,3}$.

Lemma 1. *A single permutation π of the alphabet, applied simultaneously in every coordinate, preserves lines, line-freeness, cardinality and coordinate-subcube bounds. It permutes the three first-coordinate slice sizes arbitrarily.*

Proof. Apply π to each constant of a wildcard pattern, leaving its wildcard positions unchanged. The image of $w(a)$ is the resulting pattern evaluated at $\pi(a)$. Since π is bijective, the image of the three-point line is exactly a line. The same map is bijective on all words and carries coordinate subcubes to subcubes of the same dimension. A first-coordinate slice labeled a becomes the slice labeled $\pi(a)$. Choosing π to send the three slice labels to their order by size proves the last assertion, including ties. □

Permuting symbols in the first coordinate alone is not the symmetry used here. For example, swapping 0 and 1 in that coordinate sends the line $\{00, 11, 22\}$ to $\{10, 01, 22\}$, which is not a combinatorial line. Earlier proof descriptions used this invalid local explanation. Lemma 1 corrects it; the case formulas and the case cover are unchanged.

3 The finite reduction

The three auxiliary obligations are denoted L_1, L_4, L_5. They exclude line-free sets of sizes at least 3, 53 and 151 in dimensions 1, 4 and 5, respectively. Their formula inputs are:

Obligation	Dimension	Required size	Proper-subcube caps used
L_1	1	3	none
L_4	4	53	$2, 6, 18$
L_5	5	151	$2, 6, 18, 52$

The refutations establish $c_1 \le 2$, then $c_4 \le 52$, then $c_5 \le 150$, in that order. Between them, (1) establishes $c_2 \le 6$, $c_3 \le 18$ and $c_6 \le 450$. There is no use of a cap of the dimension currently being proved. Thus no published small-dimensional optimality result is needed as an unchecked premise.

Suppose a line-free $A \subseteq \{0,1,2\}^7$ has $|A| \ge 1303$. Write $a_v = |A \cap \{x_1 = v\}|$. Lemma 1 permits $a_0 \ge a_1 \ge a_2$. Since each slice is six-dimensional,

$$435 = \lceil 1303/3 \rceil \le a_0 \le 450.$$

For each integer s in this interval the main formula encodes line-freeness, size at least 1303, the proper-subcube caps $2, 6, 18, 52, 150, 450$, and

$$a_0 = s \ge a_1 \ge a_2.$$

All sixteen formulas are unsatisfiable. Every counterexample to the upper bound would, after the global relabeling, satisfy one of them. Their refutations therefore imply $c_7 \le 1302$.

4 Encoding and independent semantic audit

Point $x = (x_1, \ldots, x_n)$ has variable index $1 + \sum_i x_i 3^{n-i}$, interpreted as $x \in A$. Every line contributes the clause consisting of its three negated point variables. Hierarchical unary totalizers count points in coordinate subcubes and in intermediate unions of two child subcubes. Auxiliary variable (S, j) means $|A \cap S| \geq j$, where S is a product of nonempty subsets of the alphabet. Thresholds are truncated only beyond an available cap; the first forbidden threshold is explicitly false. A root threshold forces $|A| \geq K$. Units force $a_0 = s$ and threshold implications force the slice order. The encoder and independent auditor are supplied as `checks/encode.py` and `checks/reduction.py`.

For clarity, the audit checks the clause-family conditions used in the written upper-bound argument: every hypothetical counterexample extends to a satisfying assignment by the stated threshold interpretation. It does not rely on a solver's interpretation of auxiliary variables. For a box $S = M_1 \times \cdots \times M_n$, let d be its number of unrestricted coordinates. If $d < n$, let $\bar{c}_d$ be the cap proved earlier in the chain of Section 3 (with $\bar{c}_0 = 1$), as in the auditor's `proved_table`. Since $\bar{c}_d \geq c_d$, disjoint coordinate subcubes give the valid bound

$$U(S) = \bar{c}_d \prod_{i:\,|M_i|<3} |M_i|.$$

For the entire ambient cube use the trivial bound 3^n, so the current claim is never its own premise. The empty set has bound zero.

The auditor recognizes every clause as one of the following valid statements under that interpretation.

1. A line clause forbids its three points together.
2. An upward counting clause has the form $(S, k) \vee \bigvee_r \neg(P_r, j_r)$, where the P_r are disjoint subsets of S and $\sum_r j_r \geq k$. Its validity follows by adding counts.
3. A downward counting clause has the form $\neg(S, k) \vee \bigvee_r (P_r, j_r)$. The remainder $S \setminus \bigcup_r P_r$ is empty or a box; disjointness and containment of the P_r in S are not required. By the union bound, when all positive literals are false,

$$|A \cap S| \leq \sum_r \min(j_r - 1, U(P_r)) + U(S \setminus \textstyle\bigcup_r P_r) \leq k - 1,$$

   which proves the clause.
4. A cap unit $\neg(S, k)$ has $U(S) \leq k - 1$.
5. A premise clause follows from $|A| \geq K$, $a_0 = s \geq a_1 \geq a_2$: these include applicable positive root and slice thresholds, negative slice thresholds beyond s, and order implications $\neg(a_v \geq t) \vee (a_u \geq t')$ for $u < v$ and $t' \leq t$.

The product sets and thresholds are read from an encoder-generated semantic map, but the auditor independently checks point indexing, the inclusion and disjointness conditions for upward clauses, bounds and every clause. The map supplies meanings, not evidence of their validity. The audit also checks all line clauses are present, the size and profile premises are present, header counts agree, and the exact integer case cover is complete. For the main formulas the rule counts are 14,197 line clauses, 940,716 upward clauses, 940,644 downward clauses, 765 cap clauses and 903 premise clauses (902 at $s = 450$, where a cap already supplies one unit).

The logical dependency is therefore explicit: the written validity of these rules and the cap induction, the exhaustive clause audits, and checked unsatisfiability together rule out each set-theoretic counterexample. These steps have not been formalized together in a proof assistant.

5 Refutations and the lower bound

CaDiCaL [15] generates text LRAT proofs [16]. The checking tools are FRAT-rs [18], drat-trim's standalone LRAT checker [17], and cake_lpr [19], built using CakeML [20]. The source-revision prefixes are CaDiCaL `c6073042` (version 3.0.1), FRAT-rs `aa185814`, cake_lpr `a36874a8`, and drat-trim `2e3b2dc0`. Full source pins and build commands are in `checks/README.md`; the CaDiCaL tool-paper citation describes the solver family, not a claim that version 2.0 produced these proofs. CaDiCaL is built with `./configure` and `make`, FRAT-rs with `cargo build -release -locked`, and the drat-trim source with `cc -O2`. The cake_lpr build compiles the upstream pre-generated assembly and C runtime, using `make cake_lpr_arm8` on ARMv8 or `make cake_lpr` on x86-64. These are the upstream verified checker's build targets, not a new formal verification of our reduction. Its guarantee is relative to the formal DIMACS and LPR parser specifications, HOL4 and the modeled execution environment [19]; it establishes unsatisfiability of the parsed formula.

The producer options are `-q -lrat=true -binary=false`; the LRAT stream is checked directly without proof conversion. The following table records the historical coverage, with each cell giving the campaign and whether the recorded CNF SHA-256 matched the audited formula. P denotes the original production campaign, I the independent regeneration of all nineteen proofs, and C the later eight-case cake-only regeneration. Each grouped row applies to every integer s in the stated interval.

Obligations	drat-trim	FRAT-rs	cake_lpr
L_1	P / matched	I / matched	I / matched
L_4	P / matched	I / matched	I / matched
L_5	P / matched	I / matched	I / matched
$s = 435, \ldots, 442$	P / matched	I / matched	C / matched
$s = 443, \ldots, 450$	P / matched	I / matched	I / matched

The original production ledger records all nineteen CNF hashes; they match the corresponding hashes in the independent-check records and the semantic audits. The original drat-trim binary hash is unrecorded. These formula-identity matches do not make the campaigns a single checking execution. In campaign I, the eleven cake_lpr checks shared the producer stream relayed to FRAT-rs. Its contemporaneous records retain the complete producer-stream SHA-256 and byte count, as well as the CNF SHA-256.

Campaign C regenerated the eight largest main proofs, $s = 435, \ldots, 442$, and checked each with cake_lpr alone. All eight returned `s VERIFIED UNSAT` with checker exit zero and solver exit 20. Each audited CNF matched its contemporaneously archived independent-check SHA-256. A relay hashed and counted the complete, unmodified producer stream while offering it through a FIFO to cake_lpr; its SHA-256 and byte count matched that archived stream. This identifies the proof supplied for checking. It does not assert that cake_lpr parsed every trailing byte after accepting an empty-clause derivation. This campaign added no FRAT-rs or drat-trim checks. The successful cake_lpr heaps for these eight cases ranged from 32,768 to 60,000 MiB.

A full public replay requires FRAT-rs, the supplied drat-trim LRAT checker and cake_lpr to accept every regenerated obligation. An UNSAT exit code alone never passes: each checker must accept an empty-clause derivation. Public replay acceptance depends on proof validity for the audited CNF, not on reproducing a historical stream hash; byte-identical regenerated output across builds is not promised. The large proof streams are not distributed. One complete set of nineteen proofs comprises 1,557,693,107,853 bytes (about 1.56 TB), consumed through pipes.

The historical independent computation used about 64 solver CPU-hours, 9.7 FRAT-rs CPU-hours and 0.9 cake_lpr CPU-hours for its eleven cake_lpr checks, plus encoding and audits; its slowest case took about 21.4 hours elapsed. Separately, the eight additional cake-only checks regenerated 1,457,126,863,436 bytes (about 1.46 TB) and used 100.7 CPU-hours in total: 68.0 for the solver, 25.1 for cake_lpr and 7.6 for stream handling and audits. Their slowest case took 24.9 hours elapsed. These additional solver and cake_lpr costs do not represent further historical FRAT-rs or drat-trim checks. A fresh public run also pays for drat-trim checking. These are measured costs, not a promise for other hardware. The quick check audits the reduction and the witness and runs a small proof with corruption controls; it does not replay the sixteen main refutations.

For the lower bound, the supplied 1302 words (written over $\{1, 2, 3\}$) are a realization of Polymath1's known lower bound [3, Section 2]. They consist of all words with letter-count triples

$$034, 043, 124, 142, 223, 232, 250, 304,$$
$$313, 331, 340, 412, 421, 502.$$

Their multinomial weights sum to 1302. Two independently written witness checkers verify distinctness, cardinality and absence of all 14,197 combinatorial lines. We do not identify this particular list with a specific source construction or claim a new lower bound. Together with the upper bound this proves Theorem 1.

Availability and license. Versions 1–4 and their artifacts are archived at [11, 12, 13, 14]. The current checking commands and dependencies are in the accompanying README and `checks/README.md`. The accompanying bundle contains the manuscript source, the explicit witness, the formula specification, the encoder, semantic and symmetry audits, witness checkers, a standard LRAT checker and instructions for pinned external tools. Large proofs are regenerated. The work is licensed under Apache-2.0; the bundled drat-trim checker retains its MIT notice. See `THIRD_PARTY.md` for dependencies and attribution.

References

[1] H. Furstenberg and Y. Katznelson, *A density version of the Hales–Jewett theorem*, J. Anal. Math. 57 (1991), 64–119.

[2] D.H.J. Polymath, *A new proof of the density Hales–Jewett theorem*, Ann. of Math. 175 (2012), 1283–1327, doi:10.4007/annals.2012.175.3.6.

[3] D.H.J. Polymath, *Density Hales–Jewett and Moser numbers*, in An Irregular Mind, Bolyai Society Mathematical Studies 21, Springer, 2010, pp. 689–753, arXiv:1002.0374v2; doi:10.1007/978-3-642-14444-8_22.

[4] T. Tao, *DHJ(3): 900–999 (Density Hales-Jewett type numbers)*, Polymath discussion post, March 4, 2009, https://terrytao.wordpress.com/2009/03/04/dhj3-900-999-density-hales-jewett-type-numbers/.

[5] K. O'Bryant, *Sets of Natural Numbers with Proscribed Subsets*, J. Integer Seq. 18 (2015), Article 15.7.7, https://cs.uwaterloo.ca/journals/JIS/VOL18/OBryant/obr3.pdf.

[6] OEIS Foundation, A156989: density Hales–Jewett numbers for an alphabet of size 3.

[7] Y. Mouhib, *Lower Bounds for the Hales–Jewett Numbers via Symmetric and One-Weight Colorings*, arXiv:2606.22155v2, 2026.

[8] N. Conlon, N. Farnsworth and A. Robertson, *On Hales–Jewett and Related Numbers*, arXiv:2607.18111v1, 2026.

[9] S. Shelah, *HJ numbers revisited*, arXiv:2607.14732v3, 2026.

[10] G. Dahia, *DensityHalesJewett*, formalization of the Dodos–Kanellopoulos–Tyros proof, `https://github.com/gdahia/DensityHalesJewett`, 2026.

[11] P. Graham, *The seventh ternary density Hales–Jewett number*, version 1, posted October 20, 2026, doi:10.5281/zenodo.23491269.

[12] P. Graham, *The seventh ternary density Hales–Jewett number*, version 2, posted October 20, 2026, doi:10.5281/zenodo.23509560.

[13] P. Graham, *The seventh ternary density Hales–Jewett number*, version 3, posted October 21, 2026, doi:10.5281/zenodo.23532529.

[14] P. Graham, *The seventh ternary density Hales–Jewett number*, version 4, posted October 22, 2026, doi:10.5281/zenodo.23572685.

[15] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks and F. Pollitt, *CaDiCaL 2.0*, CAV 2024, LNCS 14681, pp. 133–152, doi:10.1007/978-3-031-65627-9_7.

[16] L. Cruz-Filipe, M. J. H. Heule, W. A. Hunt Jr., M. Kaufmann and P. Schneider-Kamp, *Efficient Certified RAT Verification*, CADE 2017, LNCS 10395, pp. 220–236, doi:10.1007/978-3-319-63046-5_14.

[17] N. Wetzler, M. J. H. Heule and W. A. Hunt Jr., *DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs*, SAT 2014, LNCS 8561, pp. 422–429, doi:10.1007/978-3-319-09284-3_31.

[18] S. Baek, M. Carneiro and M. J. H. Heule, *A Flexible Proof Format for SAT Solver-Elaborator Communication*, TACAS 2021, LNCS 12651, doi:10.1007/978-3-030-72016-2_4.

[19] Y. K. Tan, M. J. H. Heule and M. O. Myreen, *cake_lpr: Verified Propagation Redundancy Checking in CakeML*, TACAS 2021, doi:10.1007/978-3-030-72013-1_12.

[20] R. Kumar, M. O. Myreen, M. Norrish and S. Owens, *CakeML: A Verified Implementation of ML*, POPL 2014, pp. 179–191, doi:10.1145/2535838.2535841.