CERTIFIED THREE-COLOUR RADO NUMBERS FOR $ax + by = cz$ WITH NON-UNIT COEFFICIENTS

KOREA SUPERINTELLIGENCE LABS

ABSTRACT. For positive integers a, b, c and $r \geq 1$, the Rado number $R_r(ax + by = cz)$ is the least N such that every r-colouring of $\{1, \ldots, N\}$ admits a monochromatic solution of $ax + by = cz$. Several hundred values of R_3 are in print, and those not covered by a closed form are the output of a satisfiability solver reporting an unaudited verdict; as far as our search reached, no proof log accompanies any of them. We close that gap for seven equations. For each of

$$5x + 3y = 3z,\ 3x + 3y = 2z,\ 5x + y = z,\ 5x + 3y = 2z,\ 6x + 3y = 2z,\ 5x + 5y = 4z,\ 3x + 2y = z$$

we determine R_3 exactly — the values are 186, 243, 286, 395, 648, 875 and 1093 — and both halves of each determination are machine-checked: the avoiding colouring of $[1, R_3 - 1]$ is exhibited and verified, and the non-existence of an avoiding colouring of $[1, R_3]$ is an LRAT refutation replayed by a formally verified checker against a propositional formula that is constructed, not read from a file. The bridge between the formula and the colouring problem is a single encoding-fidelity theorem, proved once for all (a, b, c, r, N), and it is proved without any appeal to computation. The largest instance refutes a formula with 298,663 clauses. As far as our search reached, these are the first Rado numbers with coefficients other than $a = b = c = 1$ to carry a machine-checkable certificate. All statements are verified in Lean 4.

1. INTRODUCTION

Let a, b, c be positive integers and let E denote the equation $ax+by = cz$. For $r \geq 1$ write $[1, N] = \{1, 2, \ldots, N\}$. A colouring $\chi\colon [1, N] \to \{0, \ldots, r-1\}$ *avoids* E if there is no triple $(x, y, z) \in [1, N]^3$ with

$$\chi(x) = \chi(y) = \chi(z) \quad \text{and} \quad ax + by = cz.$$

The three variables are not required to be distinct; this is the standard convention for Rado numbers, and the variant that forbids repetitions is a different sequence (the weak Schur numbers). The *r-colour Rado number* $R_r(E)$ is the least N for which no colouring of $[1, N]$ avoids E. Rado's theorem [6] decides for which E this number is finite, and $R_r(x + y = z)$ is one more than the classical Schur number [7]; the off-by-one is live in the literature and in the standard tables, so we fix the convention explicitly in Section 2.

The three-colour case of $ax + by = cz$ has become a computational subject. Chang, De Loera and Wesley [2] tabulate several hundred cells of $R_3(ax + by = cz)$ together with closed forms for some infinite subfamilies, and Ahmed, Zaman and Bright [1] extend two slices of the table much further: 450 cells of $R_3(ax + by = bz)$ for $1 \leq a \leq 30$, $1 \leq b \leq 15$, all of them finite and the largest 31,711, and 375 cells of $R_3(ax + ay = bz)$ for $1 \leq a \leq 15$, $1 \leq b \leq 25$, of which 247 are finite and the largest is 97,875. For the tabulated cells the method is the same in both papers: encode "$[1, N]$ has an avoiding r-colouring" as a propositional formula, hand it to a SAT solver, and record the value of N at which the answer turns from satisfiable to unsatisfiable.

The lower half of such a determination — that $[1, N-1]$ *does* have an avoiding colouring — comes with a witness, and any reader can check it in quadratic time. The upper half does not. It is a claim that a search over a space of size r^N found nothing, and in the published record it is supported by nothing beyond the solver's word. Neither of the two papers reports a proof log. The proof of the computational theorem of [2] says only that the authors "verified using a SAT solver that the formula … is unsatisfiable"; the strings DRAT, LRAT, `drat-trim` and `cake_lpr` occur nowhere in either paper, and the two occurrences of "certificate" in [1] both name a satisfying assignment —

one of them in so many words, "certificates (satisfying assignments) generated by the SAT solver CaDiCaL". This is exactly the situation that proof logging was invented for, and for the equation $x + y = z$ it has already been dealt with: Heule's determination of the fifth Schur number [3] produced a DRAT proof, converted it to LRAT, and had it certified by a formally verified checker. For every other equation the gap is open.

Two recent systems bring solver certificates into the Lean 4 proof assistant by reflection, and both settle a Schur number with one. LRAT-Catcher [4] runs Lean core's formally verified LRAT checker as compiled native code and reports the Schur number $S(4) = 44$ and the Ramsey number $R(4, 4) = 18$ as Lean theorems. LeanCSP [5] proves reformulations of constraint problems — equivalence, equisatisfiability, soundness of symmetry breaking — parametrically for families that include n-queens, Latin squares, Schur triples, and Ramsey and van der Waerden numbers, and closes the loop with pseudo-Boolean rather than LRAT certificates: its $S(4) = 44$ comes from a VeriPB proof checked inside Lean. Neither treats $ax + by = cz$ with coefficients other than $a = b = c = 1$ — the string "Rado" occurs in neither — and both of the published Rado computations cited above predate them.

What is settled here. We give seven equations with non-unit coefficients for which R_3 is determined, in full, inside a proof assistant. The values are

$$R_3(5x + 3y = 3z) = 186, \qquad R_3(3x + 3y = 2z) = 243, \qquad R_3(5x + y = z) = 286,$$
$$R_3(5x + 3y = 2z) = 395, \qquad R_3(6x + 3y = 2z) = 648, \qquad R_3(5x + 5y = 4z) = 875,$$
$$R_3(3x + 2y = z) = 1093.$$

None of these numbers is claimed as new, and every one of the seven was checked against the published tables cell by cell. All seven lie in the range $1 \le a, b, c \le 6$ covered by the six tables of $R_3(ax+by = cz)$ in [2], and four of them — $5x+3y = 3z$, $5x+y = z$, $3x+3y = 2z$ and $5x+5y = 4z$ — lie in the two tables of [1] as well. Every value proved below agrees with the published entry for its own equation, and with the published entry for the equations obtained from it by scaling (a, b, c), in every table that carries one — with two exceptions, both in the $R_3(ax + by = 4z)$ table of [2]. That table gives 141 at $5x + 5y = 4z$, where [1] and the certificate below both give 875; and it gives 31 at $6x + 6y = 4z$, where [1] gives 243 and where the $R_3(ax + by = 2z)$ table of [2] itself gives 243 for the equivalent equation $3x + 3y = 2z$, which is also the value certified below. No value in this paper was read off a table — each was located by an independent search and then proved — so no theorem here depends on a table having been transcribed correctly, and where the printed tables disagree the certificate is what decides. What is new is the certificate. Each of the seven upper halves is an LRAT refutation, produced by an off-the-shelf solver and then replayed inside Lean against a formula that a Lean function *builds*; the connection between that formula and the colouring problem is a single theorem (Theorem 3.1) proved for all parameters at once, with no computation in it. As far as the search recorded in Section 4.1 reached, no Rado number with coefficients other than $a = b = c = 1$ previously carried a machine-checkable refutation.

We do not touch $r \ge 4$. Four colours are barely charted: [2] gives a single table of $R_4(a(x-y) = bz)$ with thirteen entries, twelve exact values and one that is only the inequality $R_4(2(x-y) = 3z) > 225$, and [1] records that its authors "did not attempt to compute Rado numbers involving more than three colours due to the rapid growth in the SAT instances". That is where the first genuinely unsettled cells lie.

2. The threshold formulation

Fix $a, b, c \ge 1$ and $r \ge 1$, and write E for $ax + by = cz$.

Definition 2.1. For $N \ge 0$ let $\mathrm{Feas}(E, r, N)$ hold if some $\chi\colon [1, N] \to \{0, \dots, r-1\}$ avoids E. We say $R_r(E) = N$ when

$$\mathrm{Feas}(E, r, N-1) \quad \text{and} \quad \neg\,\mathrm{Feas}(E, r, N).$$

This is the "least forcing N" convention, which is the one the Rado-number literature uses. The classical Schur number is the other one: $S(r) = R_r(x+y=z)-1$, and the two conventions are both in print for the same objects — the sequences $2, 5, 14, 45, 161$ and $1, 4, 13, 44, 160$ are the same data. Everything below is stated in the first convention.

Two elementary facts make the definition well posed. The first is that avoidance is preserved by restriction: if χ avoids E on $[1, N]$ and $M \le N$, then the restriction of χ to $[1, M]$ avoids E, because a solution inside $[1, M]$ is a solution inside $[1, N]$ and the two colourings agree on it. Hence $\mathrm{Feas}(E, r, \cdot)$ is downward closed.

Lemma 2.2. *At most one N satisfies $R_r(E) = N$. Moreover $R_r(E) = N$ implies $\mathrm{Feas}(E, r, M)$ for all $M < N$ and $\neg\,\mathrm{Feas}(E, r, M)$ for all $M \ge N$.*

Proof. Immediate from downward closure of Feas: if $N < N'$ both satisfied the definition then $\mathrm{Feas}(E, r, N'-1)$ and $N \le N'-1$ would give $\mathrm{Feas}(E, r, N)$, contradicting the second half at N. □

Lemma 2.2 is what turns each determination below into a statement about a whole half-line of wrong values rather than about two neighbours, and it is the reason the negative controls of Section 5 need no second search.

The asymmetry of the two halves is the whole computational story. $\mathrm{Feas}(E, r, N)$ is settled by one colouring and a check that runs over the solutions of E inside $[1, N]$, of which there are $O(N^2)$: for each pair (x, y) the value z is determined. Its negation quantifies over all r^N colourings; for $r = 3$ and $N = 1093$ that is 3^{1093}, and no enumeration will do.

3. The encoding and the fidelity theorem

We encode "$[1, N]$ has an avoiding r-colouring" in conjunctive normal form. Let

$$S(a, b, c, N) = \big\{(x, y, z) \in [1, N]^3 \;:\; ax + by = cz\big\},$$

enumerated with x in the outer loop, y in the inner loop and $z = (ax + by)/c$ forced (a pair (x, y) contributes nothing when $c \nmid ax+by$ or when the quotient leaves $[1, N]$). The propositional variables are $v_{i,j}$ for $i \in [1, N]$ and $0 \le j < r$, read as "the integer i has colour j" and laid out at index $r(i-1)+j$. The formula $\Phi^r_{a,b,c}(N)$ is the conjunction of

- one *at-least-one-colour* clause $\bigvee_{j<r} v_{i,j}$ for each $i \in [1, N]$;
- for each $(x, y, z) \in S(a, b, c, N)$ and each colour j, the clause $\bigvee_{i \in D(x,y,z)} \neg v_{i,j}$, in which $D(x, y, z)$ is the list of the *distinct* elements of $\{x, y, z\}$, in decreasing order.

Two features are deliberate. There are no at-most-one-colour clauses: the encoding is used in one direction only, and their absence can make the solver's task harder but can never make a refutation unsound. And the deduplication in $D(x, y, z)$ is not cosmetic — when $a + b = c$ the triple $x = y = z$ is a solution and its clause collapses to a unit clause, which is the syntactic shadow of the fact that $R_r = 1$ whenever $a + b = c$.

Theorem 3.1 (Encoding fidelity)**.** *Let $\chi\colon [1, N] \to \{0, \dots, r-1\}$ avoid $ax+by = cz$ and let A be the assignment with $A(v_{i,j}) = \mathrm{true}$ if and only if $\chi(i) = j$. Then A satisfies $\Phi^r_{a,b,c}(N)$. Consequently, if $\Phi^r_{a,b,c}(N)$ is unsatisfiable then $\neg\,\mathrm{Feas}(ax + by = cz,\ r,\ N)$.*

Proof. An at-least-one-colour clause for i is satisfied by the literal $v_{i,\chi(i)}$. For a solution clause coming from $(x, y, z) \in S(a, b, c, N)$ and colour j: were all three of $\chi(x), \chi(y), \chi(z)$ equal to j, the triple would be a monochromatic solution, which χ forbids; so at least one of x, y, z — necessarily an element of $D(x, y, z)$ — has a colour other than j, and the corresponding negative literal is true. The membership fact used here is that every element of $S(a, b, c, N)$ really is an in-range solution, which is proved directly from the definition of the enumeration. The consequence follows by contraposition: an avoiding colouring would yield a satisfying assignment. □

Theorem 3.1 contains no computation and is proved once for all (a, b, c, r, N); every instance below is an application of it. Note which direction is proved. The converse — that a satisfying assignment yields an avoiding colouring — is never used and never proved, and that asymmetry is what makes the omission of the at-most-one clauses harmless. By the same token, a clause the encoder failed to emit could only weaken the formula, so unsatisfiability of the emitted formula still implies what Theorem 3.1 says it implies.

For the lower half we use the same enumeration. Given an array $c_1, \dots, c_N$ of colours, the predicate "no member of $S(a, b, c, N)$ is monochromatic under c" is decided by one walk over $S(a, b, c, N)$ with $O(1)$ array reads, hence in $O(N^2)$ steps, and it implies that the colouring it names avoids E. This direction needs the enumeration to be *complete* — every in-range solution is listed — which is also proved directly. An omission there could not produce a false theorem: it would make the check accept a non-avoiding colouring, and the implication "check passes $\Rightarrow$ colouring avoids" would then not be provable.

Remark 3.2. The clause list is built by a function inside the proof assistant rather than read from a file, so the identity of the formula the solver saw is not taken on trust: LRAT names input clauses by position, and any disagreement in the clause order, in the literal order inside a clause, or in the deduplication makes the verified checker reject the certificate rather than produce a wrong theorem. Independently of that, the constructed clause list was dumped to DIMACS and compared byte for byte against the output of the standalone generator on twelve instances, including all eight instances for which this development had previously committed a DIMACS file. All twelve agreed, headers included.

4. Seven certified values

Each cell was located outside the proof assistant, by doubling and bisection on N with the solver `cadical` [8] (version 2.1.2, as shipped with the Lean toolchain), and then proved twice over: the colouring of $[1, R_3 - 1]$ is exhibited as an explicit array and checked, and the refutation at R_3 is the solver's LRAT proof replayed by the verified checker of Section 7 against $\Phi^3_{a,b,c}(R_3)$.

Theorem 4.1. *For three colours,*

$$R_3(5x + 3y = 3z) = 186, \quad R_3(3x + 3y = 2z) = 243, \quad R_3(5x + y = z) = 286,$$
$$R_3(5x + 3y = 2z) = 395, \quad R_3(6x + 3y = 2z) = 648, \quad R_3(5x + 5y = 4z) = 875.$$

Theorem 4.2. $R_3(3x + 2y = z) = 1093$. *That is,* $[1, 1092]$ *carries a* 3*-colouring with no monochromatic solution of* $3x + 2y = z$*, and* $[1, 1093]$ *carries none. The upper half is a refutation of a formula with* 298,663 *clauses over* 3279 *variables.*

Proof of Theorems 4.1 and 4.2. Both halves of every cell rest on exhaustive machine computation, of two different kinds.

Lower half. For each cell an explicit colouring of $[1, R_3 - 1]$ is recorded as a list of $R_3 - 1$ colours. The finite search that establishes it is the walk over $S(a, b, c, R_3 - 1)$ described in Section 3: every solution triple of the equation inside the interval — from about $3 \cdot 10^3$ to about 10^5 of them, depending on the cell — is examined and found non-monochromatic. By the completeness of the enumeration this yields Feas at $R_3 - 1$.

Upper half. For each cell the formula $\Phi^3_{a,b,c}(R_3)$ is constructed inside the proof assistant, and the LRAT refutation produced by `cadical` is checked against it by a formally verified LRAT checker. Table 1 gives the size of each formula and of each certificate. The verified checker returns "true" only if every step of the certificate is a valid resolution-with-unit-propagation step ending in the empty clause, so the outcome is $\neg$Feas at R_3 by Theorem 3.1. No enumeration over colourings takes place at any point: the space has size 3^{R_3}.

The two halves together are the definition, so R_3 has the stated value. What is trusted in this argument, and what is not, is set out in Section 7. □

equation	R_3	clauses	LRAT lines	certificate	`cadical`
$5x+3y=3z$	186	10,287	6,702	814 KB	0.11 s
$3x+3y=2z$	243	19,926	3,098	464 KB	0.08 s
$5x+y=z$	286	24,397	1,641	204 KB	0.05 s
$5x+3y=2z$	395	31,364	53,402	6.92 MB	1.38 s
$6x+3y=2z$	648	70,308	26,043	3.00 MB	1.02 s
$5x+5y=4z$	875	185,150	40,347	6.02 MB	2.08 s
$3x+2y=z$	1093	298,663	16,418	2.84 MB	2.28 s

TABLE 1. The seven certified cells. "clauses" counts the clauses of $\Phi^3_{a,b,c}(R_3)$; the last column is single-core solving time, measured on a shared machine and reproducible only to within about 25%.

Corollary 4.3. *For each of the seven equations, the value in Theorems 4.1 and 4.2 is the only value R_3 can take, and every claimed value below it fails the upper half while every claimed value above it fails the lower half.*

Proof. Lemma 2.2 applied to the two halves proved above. □

4.1. **The claim we do and do not make.** The seven values are not offered as new numbers. They are offered as the first Rado numbers with coefficients other than $a=b=c=1$ whose upper half is backed by a machine-checkable certificate, and that claim rests on a literature search rather than on a proof. The search was run on 2026-08-22 and covered: a web index; the arXiv metadata API for `all:"Rado number"` (27 results, exhaustive); locally fetched and grepped full texts of [2], [1], [3] and [4]; the Semantic Scholar graph API; the OEIS JSON API; and the Archive of Formal Proofs entry pages. Query strings included `"Rado number" DRAT`, `"Rado number" LRAT proof`, `"Rado number "proof certificate"`, `"Rado number" SAT solver computing`, and `formalization Schur number Lean Coq Isabelle`. Two searches failed and are recorded as failed: the Archive of Formal Proofs keyword search (not scrapable) and the GitHub code search API (authentication required); Google Scholar was reached only indirectly. [5] appeared after that search and was read the same way the following day.

The search also corrected two claims we had previously made. A machine-checkable certificate for *some* Rado number does exist — $R_r(x+y=z)$ is a Rado number, and [3] certified the $r=5$ case — so the unqualified form of the claim is false. And the mechanism used here, a verified LRAT checker run by reflection inside Lean 4, is prior art [4]. What survives is the narrow statement above, with "not found" understood as not found rather than as not existing.

5. Controls

An encoding that is wrong in the right way can produce theorems that typecheck and say nothing. Five failure modes are ruled out explicitly, each by a statement proved alongside the results.

(1) *The formula could be unsatisfiable for the wrong reason.* A mis-numbered variable would make $\Phi^r_{a,b,c}(N)$ unsatisfiable at every N, and Theorem 3.1 would then prove every Rado number to be 1 while every proof still went through. Each cell's own lower half refutes this: an avoiding colouring of $[1, R_3-1]$ is a satisfying assignment of $\Phi^3_{a,b,c}(R_3-1)$ by Theorem 3.1, so that formula is satisfiable and no certificate for it can exist. This is recorded for all seven cells.

(2) *The variable layout could drift* from the one the solver read. Beyond the byte-level comparison of Section 3, the three values that fix the layout (the first index, the last colour of one integer's block, the start of the next block) are pinned, as are the clause counts of the four smaller formulas against the DIMACS header the solver read.

(3) *The literal order inside a solution clause* is the one part of the encoding that is not forced, and the deduplication is load-bearing. All four shapes a triple can take (x, y, z distinct; two equal, low or high; all three equal) are pinned, including the unit clause produced in the last case.
(4) *The interval could be wrong.* The whole subject collapses if $[1, N]$ is read as $\{0, \dots, N-1\}$, since $0 + 0 = 0$ is then a monochromatic solution of $x + y = z$ at every size and every Rado number becomes 1. The enumeration is checked to contain $(1, 1, 2)$, to be non-empty, and to contain no triple with an entry 0; and for $x + y = z$ at $N = 5$ it is checked to have exactly 10 members, against the 125 triples of $[1, 5]^3$, so it is selecting solutions rather than everything.
(5) *The witness check could be vacuous.* If the $O(N^2)$ check accepted every array, the lower half of every cell would be free. A constant colouring and an empty array are both rejected, and the sum-free colouring $\{1, 4\} \mid \{2, 3\}$ of $[1, 4]$ is accepted.

One further check is an agreement rather than a refutation. Two earlier certificates in this development, for $R_3(x + y = z) = 14$ and $R_3(4x + 3y = 3z) = 109$, were produced for a different import route — one that compiles a certificate into an explicit proof term, with a separate generated lemma per clause, and which reaches about 4500 clauses before exhausting memory. Both are replayed here through the verified checker against the *constructed* formula, and both go through, so the two routes agree on the same two values. Since a single byte of disagreement between the constructed formula and the DIMACS file the solver read would make the checker reject, this is a byte-level cross-check of the encoding as well.

6. Cost, and where the method stops

The following are measurements on one shared machine (load average 25–35 throughout), not theorems, and should be read to within about 25%. All seven cells of Table 1, checked in the proof assistant, cost about 20 seconds and 0.34 GiB above the baseline cost of loading the development. For comparison, the proof-term import route mentioned in Section 5 costs 352 seconds and 14.9 GB for five *smaller* cells, none above $R_3 = 109$. The largest cell here, at 298,663 clauses, is about 66 times that route's measured clause ceiling.

Two cells beyond the seven were solved, bridged and checked standalone but are not reported as theorems: $R_3(6x + 5y = 3z) = 1074$ (173,130 clauses, a 22.4 MB certificate, 14.4 s of solving) and $R_3(5x + 5y = 3z) = 2000$ (722,600 clauses, an 88.7 MB certificate, 44.9 s of solving, and 25.9 s at a peak of 4.12 GiB to check). The second is where this run stopped: its resident memory is about ten bytes per certificate byte, and it breaches the 4 GiB budget we imposed per check. Three limits bind, in this order — a certificate small enough to archive ($\approx$ 25 MB, around $R_3 \approx 1100$), memory (4 GiB, around $R_3 \approx 2000$), and the machine itself (a 6 GB certificate), which nothing here approached. The solver is none of them: it still finishes in under a minute at $R_3 = 2000$. The largest cells of the published tables are far out of reach of this method as implemented: the number of solutions of $ax + by = cz$ inside $[1, N]$ is $\Theta(N^2)$, and each contributes three clauses, so $R_3(30x + y = z) = 31{,}711$ and $R_3(15x + 15y = 8z) = 97{,}875$ — the largest finite entries of the two tables of [1] — call for formulas with something like 10^7 and 10^8 solution triples respectively.

Remark 6.1. A sweep of $R_3(ax + by = cz)$ over $1 \le a, b, c \le 6$ with the same solver, each value located by doubling and bisection rather than read from a table, found 33 cells with $R_3 > 109$; the largest are $5x + 5y = 3z$ at 2000, $3x + 2y = z$ and $6x + 4y = 2z$ at 1093, and $6x + 5y = 3z$ at 1074. Every one solves in under a minute. This computation lies outside the formal development: it was used to choose targets, and nothing in this paper depends on it.

Remark 6.2. The cheap half is what runs out first, repeatedly. Before the $O(N^2)$ witness check of Section 3 was written, the same four smallest cells cost 131 seconds and peaked at 7.99 GiB with a kernel-evaluated witness check that walks $\Theta(N^2)$ pairs but spends a modular exponentiation at each step, hence $\Theta(N^3)$ bignum operations; the $R_3 = 1093$ cell with that check was killed after ten minutes, while its certificate side takes eleven seconds. In this development the witness side

has become the binding constraint three times, each time immediately after the certificate side got cheaper.

7. Formal verification

Every statement above carrying a theorem number is proved in Lean 4 [9] against Mathlib [10], and the proofs are part of a repository whose reference is to be added at submission. Theorem 3.1, the restriction and uniqueness lemmas of Section 2, the soundness and completeness of the solution enumeration, the variable-layout pins of item 2 of Section 5 and the controls of items 3–5 are ordinary proofs: their axiom sets are exactly `propext`, `Classical.choice` and `Quot.sound`, the three axioms of Lean's logical foundation, and no appeal to compiled code enters them. Theorems 4.1 and 4.2 additionally evaluate two Boolean programs on concrete data by compiled native code: Lean core's formally verified LRAT checker, applied to the certificate and the constructed formula, and the $O(N^2)$ witness check, applied to the recorded colouring. So do the clause counts of item 2, the control of item 1 — which reuses each cell's own witness — and the two replayed certificates at the end of Section 5. Each such evaluation adds the axiom `Lean.ofReduceBool`, which places the Lean compiler in the trusted base for those theorems — the standard cost of native reflection, and the same one paid by [4]. The LRAT checker itself is not trusted: it is verified inside Lean, and the solver that produced the certificates is not trusted at all, since a wrong certificate makes the checker return false. The certificates and the explicit colourings are archived with the sources.

Acknowledgements

References

[1] T. Ahmed, L. Zaman, and C. Bright. *Symbolic Sets for Proving Bounds on Rado Numbers*. In *Proceedings of the 10th International Workshop on Satisfiability Checking and Symbolic Computation (SC-Square 2025)*, CEUR Workshop Proceedings 4116, pages 55–75, 2025. Also arXiv:2505.12085; the tables cited here are read from v3.

[2] Y. Chang, J. A. De Loera, and W. J. Wesley. *Rado Numbers and SAT Computations*. In *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation (ISSAC '22)*, pages 333–342. ACM, 2022. Also arXiv:2210.03262v1, the version cited here.

[3] M. J. H. Heule. *Schur Number Five*. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 6598–6606, 2018. Also arXiv:1711.08076.

[4] S. Szeider. *LRAT-Catcher: Importing SAT Solver Certificates into Lean 4 by Reflection*. arXiv:2607.00815v1, 2026.

[5] P. Manrique and S. Szeider. *LeanCSP: A Framework for Certifying Constraint Reformulation and Solving in Lean*. arXiv:2607.28459v1, 2026.

[6] R. Rado. *Studien zur Kombinatorik*. Mathematische Zeitschrift 36 (1933), 424–470.

[7] I. Schur. *Über die Kongruenz* $x^m + y^m \equiv z^m \pmod{p}$. Jahresbericht der Deutschen Mathematiker-Vereinigung 25 (1916), 114–117.

[8] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, and F. Pollitt. *CaDiCaL 2.0*. In *Computer Aided Verification (CAV 2024), Part I*, Lecture Notes in Computer Science 14681, pages 133–152. Springer, 2024.

[9] L. de Moura and S. Ullrich. *The Lean 4 Theorem Prover and Programming Language*. In *Automated Deduction — CADE-28*, Lecture Notes in Computer Science 12699, pages 625–635. Springer, 2021.

[10] 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. ACM, 2020.