A Proof-Assistant-Checked Determination of the van der Waerden Number $W(3,3)$

from a derivational chain-of-thought certificate

Laurence Avent
laurence@arbitersec.com

21 July 2026

Abstract

We report a Lean 4 kernel-checked determination of the 3-color van der Waerden number: $W(3,3) = 27$. The value is classical (Chvátal, 1970); the contribution is the certificate and its form. Both directions are machine-checked from first principles on a 2-core machine: an existence theorem exhibiting an explicit 3-coloring of [1..26] with no monochromatic 3-term arithmetic progression, and a refutation theorem showing no such coloring of [1..27] exists, compiled from a single 382,964-step derivation in an 18-rule finite-domain calculus that covers the *entire* 21,048-node search — no clause learning, no symmetry breaking, and no trusted solver anywhere in the chain. The derivation is independently re-verified from its serialized bytes by two disjoint checkers (Python; zero-dependency Rust) and then compiled to 336 Lean shard modules plus a master theorem, in which every derivation step is re-proved by Lean from exactly its cited premises. A final, mechanically bridged theorem combines both directions over coloring *functions*, reducing the human audit surface from 534 enumerated axioms to two three-line definitions and one biconditional: for all n, every 3-coloring of $[1..n]$ contains a monochromatic 3-term AP iff $n \geq 27$. To the best of a documented literature search, no proof-assistant-checked determination of any 3-color van der Waerden number existed previously; the closest work verifies 2-color instances by reflective checking of solver logs. All artifacts ship with SHA-256 hashes; nothing in this paper depends on trusting the search that produced the certificate.

Contents

1 Introduction

Van der Waerden's theorem guarantees that for any r, k there is a least $n = W(r,k)$ such that every r-coloring of $\{1,\dots,n\}$ contains a monochromatic k-term arithmetic progression [1]. The small values have been computed for decades — $W(3,3) = 27$ by Chvátal [2], later instances by SAT methods [3, 4] — but computation and *certification* are different things. The largest computed value, $W(2,6) = 1132$, rests on a six-month FPGA-assisted search that produced no checkable proof object at all [4]. Modern SAT practice narrows this gap with DRAT proofs checked by independent tools [5, 6], and very recently PBLean [8] imported VeriPB pseudo-Boolean certificates [7] into Lean 4 by reflection, reporting semantic verification of the 2-color instances $W(2,3) = 9$ and $W(2,4) = 35$ among other small benchmarks.

This paper closes the analogous gap for the first 3-color instance, with a certificate of a deliberately different kind. Rather than checking a solver's log, the entire search *is* the proof: every inference the solver makes is a step in a fixed 18-rule calculus whose conclusions are computed by a kernel, the completed derivation is re-checked from bytes by two independent implementations, and Lean re-proves every step as its own lemma from exactly the premises the step cited. The result is, to the best of a documented search (Sec. 5), the first proof-assistant-checked determination of $W(3,3)$ — stated finally as one biconditional over coloring functions whose audit surface is two short definitions.

Everything here is tagged by evidence class: **[L]** Lean-kernel-checked artifact, **[T]** adversarially tested property, **[M]** measured quantity, **[D]** reviewed design. The companion systems report [10] describes the general machinery; this paper is self-contained about the mathematics and the certification chain.

2 The Result

Definition 2.1. For $n \in \mathbb{Z}$ and $c : \mathbb{Z} \to \mathbb{Z}$:

$$\mathrm{IsColoring}\, n\, c \;:=\; \forall i,\ 1 \le i \le n \to c\, i \in \{0,1,2\},$$

$$\mathrm{APfree}\, n\, c \;:=\; \forall a\, d,\ 1 \le a \to 1 \le d \to a + 2d \le n \to \neg\big(c\, a = c\,(a{+}d) \,\wedge\, c\,(a{+}d) = c\,(a{+}2d)\big).$$

Theorem 2.2 (Lean-checked; `vdw_three_three`). ***[L]*** $\big(\exists c,\ \mathrm{IsColoring}\, 26\, c \wedge \mathrm{APfree}\, 26\, c\big) \;\wedge\; \big(\forall c,\ \mathrm{IsColoring}\, 27\, c \to \neg\, \mathrm{APfree}\, 27\, c\big)$.

Theorem 2.3 (Lean-checked; `vdw33_characterization`). ***[L]*** *For every* $n \in \mathbb{Z}$: $\big(\forall c,\ \mathrm{IsColoring}\, n\, c \to \neg\, \mathrm{APfree}\, n\, c\big) \;\leftrightarrow\; 27 \le n$.

Theorem 2.3 is $W(3,3) = 27$ in one line. The witness for the left-to-right failure below 27 is the explicit coloring of [1..26]

$$0\,0\,1\,1\,0\,0\,1\,2\,1\,2\,2\,0\,2\,0\,0\,1\,0\,1\,1\,2\,0\,2\,2\,1\,2\,1,$$

realized in Lean as a decidable function and checked pointwise on all 156 progressions by `decide`. The Lean development is core toolchain only (Lean 4.10.0): no `mathlib`, no `sorry`, no `axiom` declarations (scanned mechanically across all 339 files) **[T]**.

3 The Certification Chain

3.1 Encoding

Variables $x_1, \ldots, x_n$ with domain $\{0, 1, 2\}$; for each 3-term AP $(a, a{+}d, a{+}2d) \subseteq [1..n]$ and each color j, one clause "some position of this AP avoids j," i.e. a disjunction of domain-literals. For n=27: 27 domain axioms and 507 clauses (169 APs $\times$ 3 colors); for n=26: 468 clauses. The encoder is ∼30 lines of Python **[D]**; its fidelity is ultimately discharged not by inspection of 534 printed axioms but by the bridge of Sec. 3.5, which re-derives every axiom from the two quantified definitions above.

3.2 Search that emits its own proof

The `vcot` engine [10] is a certified-propagation solver: unit propagation and conflicts on clauses are single kernel steps (rules OR_UNIT/OR_CONFLICT, added for this problem class), and case splitting is SPLIT/OR_ELIM with every dead branch ending in a derived $\bot$. Refuting n=27 explored 21,048 nodes and emitted 382,964 steps **[M]**; the two clause rules cut the certificate 4.4× from an initial 1,674,389 steps. Deliberately absent: clause learning, symmetry breaking, restarts — the certificate covers the full tree, so its soundness owes nothing to solver sophistication. (A modern CDCL solver decides this instance in well under a second *without* a kernel-checkable trail of this form; the trade is explicit and discussed in Sec. 6.)

3.3 Independent re-checking from bytes

The serialized certificate (30 MB JSON; step conclusions are not representable in the format and are recomputed by every checker) is verified by the Python checker (9 s) and by `vcot-check` — a zero-dependency Rust verifier with a private-constructor theorem type, `forbid(unsafe_code)`, and its own strict JSON parser — in 2.1 s at 183k steps/s **[M]**. The two checkers share no code and are held to exact-agreement differential fuzzing (2,197/2,197 files) in the companion report **[T]**.

3.4 Sharded compilation to Lean

A 383k-step derivation does not elaborate as one Lean file. The sharded compiler hoists large OR_ELIM branch subtrees into standalone modules — each proving *(variables) (needed outer facts) (branch hypothesis)* $\vdash \bot$ with the same one-lemma-per-step discipline — yielding 336 shard modules (chunk $\approx$ 1,500 steps) plus a master that imports them and discharges hoisted case arms by lemma application. Every shard and the master were accepted by Lean: 12,076 s wall for the shards on 2 cores under a dependency-ordered, resumable builder, 57 s for the master **[L][M]**. Sharding is a rendering of the one checked proof object, not a new trust boundary: Lean type-checks the composition, including that each shard's hypotheses match the facts in scope at its call site.

3.5 The combined theorem: shrinking the audit surface

The master theorem speaks about 27 integer variables under 534 hypotheses — faithful, but auditable only by reading the hypotheses. `Vdw33Combined.lean` closes this gap mechanically: it derives each of the 534 hypotheses from IsColoring/APfree instances (507 `omega` obligations with type-ascribed index normalization; 27 direct instantiations), obtains the function-level upper bound, proves the witness function's 156 AP-freeness facts by `decide` plus a generated integer case tree, adds two-line monotonicity lemmas, and concludes Theorems 2.2 and 2.3. Lean checks the whole bridge in ∼3.7 min **[L]**. After this step, trusting the result requires reading exactly: the two definitions, the two theorem statements, and the Lean kernel.

4 Numbers and Hashes

search nodes (refutation, full tree)	21,048
kernel steps in certificate	382,964
certificate size (JSON)	30 MB
Python / Rust checker	9 s / 2.1 s, both $\vdash \bot$
Lean shard modules / total files	336 / 339
shard build (2 cores, resumable)	12,076 s
master `vdw_3_3_upper`	accepted, 57 s
witness `vdw_3_3_lower`	accepted, 17.4 s
combined `Vdw33Combined`	accepted, $\sim$222 s
`sorry`/`axiom` scan	clean
toolchain	Lean 4.10.0 core; CPython 3.11.15; rustc 1.95.0

SHA-256 of the frozen artifacts (the proof object is the canonical anchor; `.lean` bytes pin this specific compiled rendering):

```
c86cc8c8d24484f3a06baa47d5458d9a17a9a7098e090f11354eb49ec2e694b1  vdw_3_3_27.proof.json
c38782fab8f421e034206775b6345b123e09259ab0b0f1103efb091fc82f0b9d  Vdw33Main.lean
aa057d92bdf00caf83b0f5a5fd4c1103d448ad4339f64058c88de9e0e60b29f7  Vdw33Lower.lean
f99a3f7d82c762602e8f1e604bf3ac1d62ea355d22aefb306e11984b01b19edb  Vdw33Combined.lean
6a7097b59cde7139cb88f27c4676bbdca33160cd628cab383e1c7ae272b5ffd0  vdw33_certificates_v2.zip
```

5 Related Work and Novelty Claim

$W(3,3) = 27$ is due to Chvátal [2]; SAT-based computation of van der Waerden numbers begins with Dransfield et al. [3] and culminates in the uncertified $W(2,6) = 1132$ [4]. Certified solving via DRAT [5] reached mechanized-checker scale with Schur Number Five [6]; VeriPB extends proof logging to CP-style reasoning [7]; PBLean [8] lands such certificates in Lean 4 by reflection, covering (per its evaluation tables) $W(2,3)$, $W(2,4)$, small Schur, Ramsey and Langford instances with semantic statements. Our search of this literature and general queries for formal verification of specific van der Waerden numbers found no proof-assistant-checked 3-color instance; the claim is phrased exactly so — *not aware of prior work, search documented in the artifact package* — and not as a proven negative. Methodologically the certificate differs from the reflective-checker line: it is a structural derivation whose every step Lean re-proves from its cited premises, i.e. the proof object is the reasoning itself rather than a verified verdict about a solver run. We claim this as a difference of artifact type and auditability, not superiority; reflective checking scales further (Sec. 6).

6 Limitations and Threats to Validity

Not new mathematics. The value has been known for 56 years; the contribution is certification. **Scale.** The whole-tree-as-proof design pays for auditability with size: 3.4 h of Lean checking for an instance CDCL dispatches in milliseconds, and a feasibility probe puts the analogous $S(4) = 44$ certificate beyond our current single-file or sharded Lean budget (checker-tier only). This approach does not compete with DRAT/VeriPB pipelines at scale; it occupies the small-instance, maximal-audit corner. **Schema trust.** The 18 rule schemas are human-reviewed and adversarially tested, not themselves formalized; every *instance* in this certificate is Lean-re-proved, so schema error cannot have produced a false theorem here, but schema-level formalization remains the right next step. **Bridge generation.** The combined theorem's 534 discharge obligations are generated by 200 lines of Python; a bug there makes Lean reject, not wrongly accept (fail-closed), and the final statements are hand-readable. **Environment.**

Single 2-core machine, single successful build (plus one OOM-killed and one suspension-killed attempt, both recorded in the artifact log); timings are indicative only. **Non-canonical shard bytes.** Recompiling the same proof object reproduces the `.lean` files today, but compiler evolution would change their hashes while proving the same statements; the proof-object hash is the stable anchor.

7 Reproduction

From the `vcot` repository (artifact package `vdw33_certificates_v2.zip`):

```
# re-solve and re-emit the certificate (engine, ~5 min)
PYTHONPATH=src python3 -c "from vcot.ramsey_problems import vdw; \
  from vcot.engine import solve; print(solve(vdw(3,3,27)).verdict)"

# re-check the shipped certificate from bytes
./rust/target/release/vcot-check artifacts/vdw33/vdw_3_3_27.proof.json
PYTHONPATH=src python3 -m vcot check artifacts/vdw33/vdw_3_3_27.proof.json

# rebuild + re-check all Lean modules (resumable; ~3.4 h on 2 cores)
python3 scripts/build_shards.py artifacts/vdw33/lean_pkg --jobs 2

# the combined axiomatic theorem
python3 scripts/gen_combined.py
LEAN_PATH=artifacts/vdw33/lean_pkg lean artifacts/vdw33/lean_pkg/Vdw33Combined.lean
```

Acknowledgments

Implementation, experiments, and drafting were carried out with Claude (Anthropic). For this result the proposer was the deterministic engine, not a language model; the assistant's role was building and operating the pipeline. The Lean community's core toolchain made kernel-checked certificates practical on modest hardware.

References

[1] B. L. van der Waerden. Beweis einer Baudetschen Vermutung. *Nieuw Archief voor Wiskunde*, 15:212–216, 1927.

[2] V. Chvátal. Some unknown van der Waerden numbers. In *Combinatorial Structures and Their Applications*, pages 31–33. Gordon and Breach, 1970.

[3] M. R. Dransfield, V. W. Marek, and M. Truszczyński. Satisfiability and computing van der Waerden numbers. *Electronic Journal of Combinatorics*, 11(1), 2004.

[4] M. Kouril and J. L. Paul. The van der Waerden number $W(2,6)$ is 1132. *Experimental Mathematics*, 17(1):53–61, 2008.

[5] N. Wetzler, M. J. H. Heule, and W. A. Hunt Jr. DRAT-trim: efficient checking and trimming using expressive clausal proofs. In *SAT*, 2014.

[6] M. J. H. Heule. Schur number five. In *AAAI*, 2018.

[7] J. Elffers, S. Gocht, C. McCreesh, and J. Nordström. Justifying all differences using pseudo-Boolean reasoning. In *AAAI*, 2020.

[8] PBLean: pseudo-Boolean proof certificates for Lean 4. arXiv:2602.08692, 2026.

[9] L. de Moura and S. Ullrich. The Lean 4 theorem prover and programming language. In *CADE*, 2021.

[10] L. Avent. Verified chain-of-thought for finite-domain constraint reasoning: a complete implementation with machine-checked certificates. Companion report with artifacts, July 2026.

[11] L. Avent. Avent's razor: intelligence as constrained search. Companion manuscript, July 2026.