Avent's Razor: Intelligence as Constrained Search

a framework with a running implementation, measured capacity–speedup data,
and a machine-checked case study

Laurence Avent
laurence@arbitersec.com

July 2026

Abstract

Classical accounts characterize intelligence by descriptive parsimony: the shortest program consistent with observation (Solomonoff induction, Kolmogorov complexity). That account says what a good hypothesis is, not how to find one with bounded resources. Avent's Razor is an operational complement: *measure a reasoning system by the constraints it can enforce during search*, because an enforced constraint eliminates candidates before they are paid for, and because enforcement — unlike insight — is mechanically checkable. The razor has a quantitative face (constraint capacity, the bits of search space a constraint system deletes) and an architectural face (make invalid reasoning states unrepresentable, so untrusted generators, including language models, can only search inside the valid envelope).

This paper states the framework with its assumptions in the open and supports it exclusively with artifacts that exist. The evidence base: a complete implementation of verified chain-of-thought reasoning for finite-domain constraints (an 18-rule kernel whose every emitted inference is re-proved by Lean 4 from exactly its cited premises, with two independent byte-level checkers held to exact agreement by differential fuzzing); a measured relationship between exactly-computed constraint capacity κ and search speedup over 300 enumerable instances — monotone and steep, and clearly sub-exponential relative to a practical baseline, as the idealized analysis itself predicts once its assumptions are dropped; two unedited language-model sessions operating the verified loop through a typed gate; and a machine-checked case study in combinatorics: the van der Waerden number $W(3,3) = 27$, certified in both directions down to a single Lean biconditional. Claims are tagged by evidence class throughout; the strong quantitative form of the razor is stated as a conjecture, and the framework's unbuilt extensions are labeled speculation. What the evidence supports is deliberately narrower than what the framing suggests is possible — and is stated as exactly that.

Contents

1 Introduction

Two questions about a reasoning system come apart under resource bounds. *What should it believe?* has a classical answer: the shortest description consistent with the data [3, 4]. *How should it search?* does not inherit that answer; Levin-style universal search [5] is optimal only up to constants that swallow practice whole. The razor proposed here addresses the second question operationally:

> *Judge a reasoning system by the constraints it can enforce during search — because enforcement prunes the space before it is explored, and because enforcement can be checked by machine even when the generator that proposes candidates cannot be trusted at all.*

The proposal has two readings, and this paper is careful to keep their epistemic status separate. The **quantitative reading** — enforced constraints buy search reductions that grow with the constraint system's capacity — is supported here by direct measurement on one domain, in a specific, stated sense, and its strong (exponential) form is an explicitly labeled conjecture. The **architectural reading** — build reasoning systems as untrusted proposers inside mechanically enforced constraint envelopes, so that the surviving reasoning is a certificate rather than prose — is no longer a proposal at all: it is implemented end to end, its soundness properties are adversarially tested, and it has produced a proof-assistant-checked result in combinatorics [1, 2].

1.1 Evidence discipline

Every claim in this paper carries a tag: **[L]** an artifact accepted by the Lean 4 kernel; **[T]** a property exercised by automated adversarial testing; **[M]** a quantity measured in a described environment and reproducible from shipped artifacts; **[D]** a design argument over reviewed code; **[C]** a conjecture or speculation, explicitly unproven. Nothing untagged is asserted. Statements about what is *not* known — including the novelty of the case study — are phrased as the outcome of documented searches, never as proven negatives.

2 The Framework

Definition 2.1 (Constraint system and capacity)**.** For a finite candidate space $\mathcal{P}$ and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of decidable predicates on $\mathcal{P}$, the feasible set is $\mathcal{P}_\mathcal{C} = \{p \in \mathcal{P} : \forall i,\ C_i(p)\}$, and the *constraint capacity* is

$$\kappa(\mathcal{C}) \;=\; \log_2 \frac{|\mathcal{P}|}{\max(1, |\mathcal{P}_\mathcal{C}|)} \quad \text{bits.}$$

Capacity measures how much of the space the constraints delete. It deliberately says nothing about the *cost* of enforcing them, nor about how much of the deletion a given search procedure actually harvests; those are properties of algorithms, not of constraint systems, and conflating them is the standard way to overclaim in this area.

Proposition 2.2 (Idealized pruning gain)**.** *Assume:* (A1) *the baseline enumerates $\mathcal{P}$ exhaustively at unit cost per candidate;* (A2) *constrained search enumerates exactly $\mathcal{P}_\mathcal{C}$, paying enforcement overhead at most a constant factor $\omega \geq 1$ per candidate;* (A3) *enforcement is exact — no false accepts or rejects;* (A4) *neither procedure reorders, propagates, or learns. Then the cost ratio is exactly $2^{\kappa(\mathcal{C})}/\omega$.*

Proof. $|\mathcal{P}|/(\omega\,|\mathcal{P}_\mathcal{C}|) = 2^{\kappa(\mathcal{C})}/\omega$ by Definition 2.1. The content of the proposition is entirely in A1–A4. □

Remark 2.3 (Why measured gains must undershoot). A1 and A4 are false for every practical baseline: real search interleaves constraint checking with enumeration, so the baseline already harvests part of κ; and real solvers propagate and reorder. Measured gains against practical baselines are therefore lower bounds on the gain over pure enumeration and can sit far below 2^κ without contradicting Proposition 2.2. The proposition is a ceiling under idealization — not a prediction — and the data of Section 4.1 should be read against exactly this remark.

Remark 2.4 (Monotonicity and redundancy). Adding constraints never shrinks capacity, and redundant constraints add zero capacity at positive enforcement cost. Constraint *selection* is therefore an economic problem; we make no claims about optimal selection here.

Remark 2.5 (Types as constraints). Under the Curry–Howard correspondence, a type system is a constraint system on the space of programs and proofs, with capacity $\kappa_{\text{type}} = \log_2(|\mathcal{P}|/|\mathcal{P}_{\text{typed}}|)$ in the sense of Definition 2.1. This is the bridge from the razor to programming-language theory: "well-typed programs cannot go wrong" is precisely "invalid states are unrepresentable, so search cannot enter them." The architectural face of the razor is this remark taken seriously for *reasoning*: reasoning steps are typed objects, a kernel computes their conclusions, and anything a generator proposes is either admitted as a checked inference or rejected with a typed error.

Definition 2.6 (Valid reasoning trace)**.** Fix a rule system R with decidable side conditions and kernel-computed conclusions. A reasoning trace is a DAG whose nodes are propositions and whose edges are R-instances; it is *valid* iff every node is an axiom or is the conclusion computed by its rule from its cited premises. A valid trace is a certificate: its acceptability depends on its bytes, not on the process that produced it.

Conjecture 2.7 (Strong quantitative razor)**.** ***[C]*** *There is a broad class of reasoning problems and a fair enumeration baseline for which enforced-constraint search achieves cost within a polynomial factor of the 2^κ idealization of Proposition 2.2.* Status: untested here. The experiment that would bear on it — a baseline ladder from pure enumeration through ground checking, forward checking, and full propagation — is specified in Section 8 and has not been run.

Conjecture 2.8 (Constraint learning)**.** ***[C]*** *Constraint systems that maximize harvested capacity per unit enforcement cost can be learned from problem distributions (outer loop: select $\mathcal{C}$; inner loop: constrained search under $\mathcal{C}$).* Status: open; no experiment here bears on it.

3 The Architecture

The architectural face of the razor prescribes four separations, all of which are implemented and exercised in the companion system, `vcot` [1]:

1. **Untrusted proposal.** Generators — language models, a deterministic engine, or a human — interact through a gate that accepts one JSON object per turn and returns either the new step id with the *kernel-computed* conclusion or a machine-readable rejection. Proposers never state conclusions; this forecloses plausible-but-unsupported claims by construction **[D]**.
2. **Small enforced calculus.** An 18-rule kernel over finite-domain integer constraints (domain facts, disequalities, offset equations, linear bounds, clause and alldifferent/Hall rules, conjunction/disjunction structure). Side conditions are finite exact computations **[D]**.
3. **Byte-level re-checking, redundantly.** Completed derivations are serialized (conclusions are not representable in the format) and re-verified by two checkers sharing no code — Python, and a zero-dependency Rust verifier whose theorem type has a private constructor (LCF discipline enforced by the compiler). The pair is held to exact verdict agreement on 2,197 valid and adversarially mutated certificates **[T]**; 1,500 mutants yielded zero semantically false acceptances against model-theoretic ground truth **[T]**.
4. **Proof-assistant finality.** Accepted derivations compile to Lean 4 with one lemma per step, each re-proved from exactly its cited premises; certificates contain no `sorry` and no `axiom` declarations, and tampering is rejected **[L][T]**. The trusted base of any final claim is the Lean kernel plus the printed problem axioms — or, where a definitional bridge exists (Section 4.3), two short definitions.

4 Evidence

4.1 Measured capacity versus measured speedup

For 300 random finite-domain instances small enough to enumerate, the capacity κ of Definition 2.1 was computed exactly by brute force, and speedup was measured as baseline search nodes over engine nodes, with the baseline defined as *the same search skeleton with propagation disabled and constraints checked once their variables are ground* **[M]**:

κ (bits)	n	geo-mean speedup	median	max
$[0, 2)$	10	1.27×	1.24×	2×
$[2, 4)$	31	2.41×	1.80×	11×
$[4, 6)$	70	8.53×	9.50×	48×
$[6, 8)$	99	19.58×	18.00×	233×
$[8, 10)$	67	43.77×	40.00×	594×
≥ 10	23	56.24×	44.00×	1,216×

Two readings, both required. The relationship is monotone and steep: more measured capacity, more measured speedup, across three orders of magnitude of instance difficulty. And the magnitudes sit far below 2^κ ($\kappa \geq 10$ bits would idealize to $\geq 1024\times$; the geometric mean is $56\times$) — consistent with Remark 2.3, since this baseline already exploits constraints on ground assignments. These data support the qualitative razor on this domain against this baseline; they neither confirm nor refute Conjecture 2.7, which needs the pure-enumeration rung of the baseline ladder.

On structured benchmarks the effect is qualitatively stronger: with certified alldifferent (Hall-set) and clause rules in the calculus, the 25-variable Einstein riddle, 4×4 and 9×9 Sudoku, and a pigeonhole refutation all close under certified propagation with *zero* search nodes, against baselines ranging from 27 to more than $2{\cdot}10^6$ nodes **[M]**. Enforced global constraints do not merely speed search here; they replace it.

4.2 Language models inside the envelope

Two unedited sessions [1], both with fresh context per call and full transcripts shipped **[M]**. A mid-tier model (Claude Haiku 4.5) solving a 6-variable puzzle committed 11 protocol violations in 28 proposals; each drew a typed, machine-readable rejection; each error class was corrected by the model in later proposals; no invalid step was ever admitted. A frontier model (Claude Fable 5) solved the full Einstein riddle through the same gate: 29 calls, 200 accepted gate actions including four model-driven case analyses, zero kernel-level rejections; the resulting 193-step chain was re-verified by both checkers and accepted by Lean in 7.1 s **[L]**. These are demonstrations of mechanics at two capability levels (n=1 each), on puzzles plausibly in training data; they support no statistical claim about model capability — and the architecture's soundness did not depend on either model behaving well.

4.3 Case study: a machine-checked determination of $W(3,3)$

Pointed at arithmetic Ramsey theory, the machinery produced what a documented literature search indicates is the first proof-assistant-checked determination of the 3-color van der Waerden number [2]: $W(3,3) = 27$. The upper bound is a 382,964-step derivation covering the entire 21,048-node search — no clause learning, no symmetry breaking, no trusted solver — compiled to 336 Lean shard modules plus a master theorem, all accepted; the lower bound is an explicit checked coloring of [1..26]; and a mechanically generated bridge combines both into one biconditional over coloring functions,

$$\texttt{vdw33\_characterization}: \quad \forall n,\ \big(\forall c,\ \mathrm{IsColoring}\, n\, c \to \neg\, \mathrm{APfree}\, n\, c\big)\ \leftrightarrow\ 27 \le n,$$

whose human audit surface is two three-line definitions **[L]**. The value is classical (Chvátal, 1970); the certification is the contribution, and its novelty is asserted only as "not found by a documented search" (closest prior work: reflective Lean checking of solver logs for 2-color instances [10]; the largest computed van der Waerden number, $W(2,6) = 1132$, has no certificate at all [11]).

For the razor, the case study carries a specific lesson: the same constraint calculus that *accelerates* search (Section 4.1) is what makes the search's outcome *checkable* — capacity and certifiability are the same machinery viewed from two sides. That identity, rather than any single speedup number, is the framework's strongest empirical support to date.

4.4 Status of claims

Claim	Status
The architecture is implementable end to end, with no trusted generator anywhere	**[L][T][M]**
Typed enforcement steers imperfect generators without soundness loss	**[M][T]** (n=1 demos)
Capacity–speedup monotonicity (this domain, stated baseline)	**[M]**
Certified global constraints can eliminate search outright (benchmarks listed above)	**[M]**
A real combinatorial value certified to proof-assistant standard	**[L]**
Strong quantitative razor (Conjecture 2.7)	**[C]**
Constraint learning (Conjecture 2.8)	**[C]**
Rule-schema soundness formalized once, in a proof assistant	open (instances are Lean-checked; schemas are reviewed
Generality beyond finite-domain constraint reasoning	open

5 Guiding Analogies (Not Results)

Two analogies inform algorithm design without carrying any load in this paper. A hard-constrained, soft-scored search posterior has Boltzmann shape, $p^*(p) \propto \mathbf{1}_{\mathcal{C}(p)}\, e^{-\beta\,\mathrm{score}(p)}$, suggesting sampling views of constrained generation; and the compression/constraint interplay resembles rate–distortion: description length trades against fit inside a feasible set. Both are stated here as analogies because that is what they are; deriving operational consequences from them is future work, not present support.

6 Directions We Deliberately Attach No Numbers To

Three directions are natural and remain entirely unbuilt **[C]**: energy-based or thermodynamic sampling over constraint-satisfying traces; geometric (e.g. hyperbolic) representations for proof-tree search; and semantics-preserving extraction of verified reasoning into deployment languages. No implementation, experiment, or theorem about any of them exists in this work, and the razor's evidential standing neither gains nor loses if they fail.

7 Threats to Validity

All measurements come from a single 2-core environment, largely single-run. The speedup baseline is one defensible choice among several, and every magnitude in Section 4.1 is relative to it; the κ sweep's instances are small by construction, and its generator has idiosyncrasies. The language-model sessions are single demonstrations on famous puzzles. The implementing assistant and the evaluated frontier proposer are the same model family (Claude Fable 5); this cannot affect soundness — the gate admits nothing invalid regardless of the proposer — but the frontier session's smoothness may benefit from protocol and prompt being co-designed by that family, and should be discounted accordingly. Rule schemas are human-reviewed and adversarially tested rather than formalized; every rule *instance* in the shipped certificates is Lean-checked. The axiom-fidelity gap — whether formal axioms mean the informal problem — is narrowed by small audit surfaces, never closed by machine.

8 Open Problems

(1) **The baseline ladder**: measure the capacity–speedup curve against pure enumeration, ground checking, forward checking, and full propagation separately; the first rung is the direct test of Conjecture 2.7. (2) **Schema-level formalization**: prove the 18 rule schemas sound in Lean once, demoting per-instance re-proof to redundancy. (3) **Sample complexity**: whether enforced constraints provably reduce the sample complexity of *learning* in this setting is open; we know of no derivation and offer none. (4) **Constraint learning**: Conjecture 2.8. (5) **A second domain**: scheduling or program synthesis, with the same evidence discipline.

9 Conclusion

The razor, stated at the strength the evidence supports: constraints that a system can *enforce* are the part of its intelligence that can be measured and checked; enforcing them yields large, monotone, measured reductions in search on the one domain tested, and yields machine-checkable reasoning as the same act; and an architecture built on this principle is not speculative — it runs, it survived adversarial testing, language models can operate it, and it has certified

a piece of mathematics down to two definitions and a biconditional. The stronger quantitative story remains a conjecture with a specified experiment. The framework earns whatever additional strength that experiment, and a second domain, deliver — and no more.

Acknowledgments

Implementation, experiments, and drafting were carried out with Claude (Anthropic); the assistant's model family is also the frontier proposer evaluated in the companion report, a dual role disclosed in Section 7. The author thanks the Lean community for a toolchain that makes kernel-checked certificates practical on modest hardware.

References

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

[2] L. Avent. A proof-assistant-checked determination of the van der Waerden number $W(3,3)$. Companion report with artifacts, July 2026.

[3] R. J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7(1):1–22, 1964.

[4] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[5] L. A. Levin. Universal search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[6] M. Gordon, R. Milner, and C. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*. LNCS 78, Springer, 1979.

[7] G. C. Necula. Proof-carrying code. In *POPL*, 1997.

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

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

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

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

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