VL-2026-05 | JUN 2026 | PROVEN-toy | $0-CPU kill-tests | not real-LLM scale

Sound Compounding: A Verifier–Frontier Ratchet for Capability Acquisition, and a Believed–vs–True Test for Self–Improvement Leakage

When does a verify-grow-reuse loop compound soundly — and how to catch a self-improvement loop deceiving itself.

Annalea Layton
Vext Labs, Inc. · alayton@tryvext.com

SCOPE · READ FIRST

All positive results are $0-CPU toy kill-tests. They validate a mechanism, not a system, and not at real-LLM scale. The real-LLM test is pre-registered and unrun. Do not cite this as having solved anything.

Vext Labs — working draft, 2026-06-29.

> **PUBLIC SCOPE (vextlabs.ai):** `PROVEN-toy` · `$0-CPU kill-tests` · **not real-LLM scale**.

Scope statement (read first): all positive results in this paper are $0-CPU toy (symbolic / small-net) kill-tests. They validate a mechanism, not a system, and NOT at real-LLM scale. The real-LLM test is pre-registered (§7) and unrun. Do not cite this as "solved" anything.

Abstract

Scaling next-token prediction yields diminishing returns: the data-scaling error exponent is empirically and theoretically pinned by the data manifold ($\alpha \approx 2m/(2m+d)$), and we confirm on a falsification battery that no architectural change we tried bends it (companion note). If capability cannot be scaled into existence, it must be acquired and accumulated. We study the conditions under which a closed verify-grow-reuse loop **compounds** capability — a capability-per-cost curve that does not diminish — rather than plateauing (the documented failure of self-improvement loops). We isolate two mechanisms and one diagnostic. **(1) Discovery via verified search:** the well-known collapse of learned routing over a skill library is avoided by searching over a self-grown certified library and admitting compositions only under a sound, different-execution-class referee; on a toy this reduces next-skill acquisition cost from $O(k^t)$ to $O(k)$ (763× cheaper at depth) with zero false admissions. **(2) Sound slice-growth, and its hard cap:** a system can author new checkers by composing certified ones, growing its verifiable region far beyond its atomic seed (reach 30, zero false admits) — but the growth is hard-capped at the closure of its seed execution class: a genuinely new execution class is never self-authorable and requires external off-support fuel. **(3) The believed-vs-true diagnostic:** self-grading loops self-deceive (believe they grew far more than they truly did); measuring believed-frontier minus true-frontier against an independent referee exposes this leakage directly (sound: gap 0, false-admits 0; self-grading: gap +1.25, 5 false admits). The naive version of the loop — reusing an approximate learned library, or one where the decomposition is handed in — is killed (it either accumulates error or is built-in-the-answer). The contribution is a precise, honest operationalization of when a verify-grow-reuse loop compounds soundly, a reusable leakage test for self-improvement claims, and a boundary theorem-in-spirit (growth = closure of the seed execution class). A real-LLM instantiation is pre-registered.

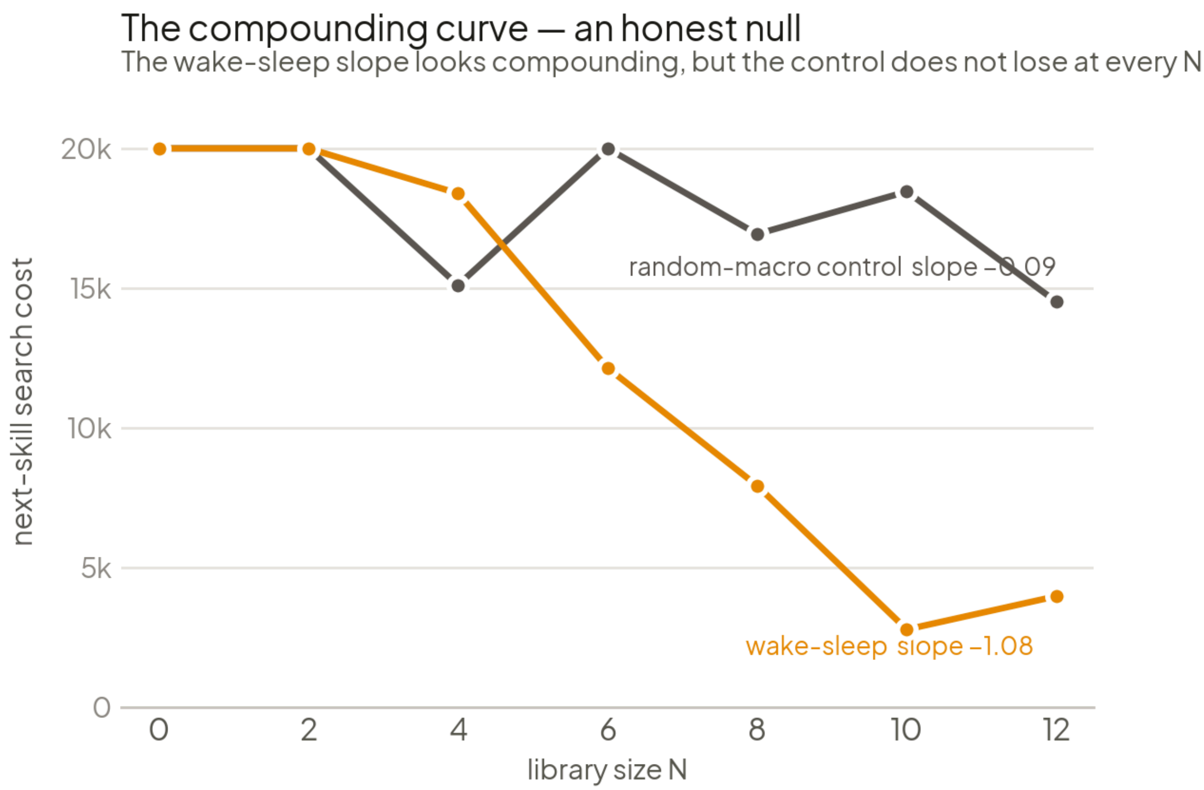
1. Introduction

Two facts frame the problem. First, scaling is a fixed diminishing curve: under passive sampling from a fixed distribution, the minimax error exponent is set by the data manifold's smoothness and intrinsic dimension, and architecture moves the prefactor, not the slope (Sharma & Kaplan; Bahri et al.; our companion falsification battery kills every architectural escape we tried, including a serial-bit-matched continuous-carry test showing the continuous inter-step channel is dense bandwidth, not new learnability). Second, reinforcement-learning-from-verifiable-rewards (RLVR) and self-improvement loops largely **re-weight** trajectories the base already samples rather than acquiring new capability (the "sharpen-only" result), and library-learning loops frequently produce single-use libraries that do not actually compound.

So the open question is not "what bigger model," but: **under exactly what conditions does a closed loop acquire-and-accumulate capability such that the marginal cost of the next capability does not grow (compounding) — soundly, without forgetting, and without self-deception?** We give toy-scale, pre-registered answers, and a leakage test that any self-improvement claim should have to pass.

2. Background and known traps (what we are not re-claiming)

- **Library learning (DreamCoder/Stitch):** MDL-driven abstraction reuse gives a compounding-flavored curve (our prior wake-sleep signal: library prior $\alpha=-1.08$ vs random -0.09) — but single-use libraries are the documented null (e.g. LEGO-Prover: 1 reuse in 189 proofs). Our discovery result is in kind a library-learning result; the contribution is the soundness gate + the bounded-cost separation, not the idea of reuse.



SOURCE: ledger/compounding_results.json (ws_beats_random_at_every_N=false). Published as a kill.

VERDICT: NO-COMPOUNDING once the 'more-tokens' artifact is subtracted

FIGURE 1 An honest null from the same program's ledger — wake-sleep next-skill search cost vs. library size N (slope −1.08) against a random-macro control (slope −0.09); verdict NO-COMPOUNDING once the more-tokens artifact is removed (`ledger/compounding_results.json`). This is the repaired control experiment that killed the naive compounding read — the same theme §4.1 revisits as "naive reuse is killed."

- **Verifier-as-filter (best-of-N + sound verifier):** real but proven — it selects, it does not create capability the base cannot sample. Our discovery uses verified search, but the creation is the per-tier verified authoring; the compounding is reuse making search cheap.
- **RLVR sharpens, does not create** (Yue et al.; ReasonMaxxer): the off-support manifold-extension must come from a different execution class (SGDF), which RLVR alone cannot supply.
- **CIP / function-preserving growth:** depth-growth with frozen old layers gives no-forgetting capacity, but growth blind to a verified signal inflates parameters without earning capability.

The novel seam is the composition: sound off-support fuel + no-forgetting fold + verified-search discovery + sound slice-growth, plus the believed-vs-true leakage test. (Independent assessment scored the mechanism novelty 5 / soundness 4 / testability 8 — a modest, honest novelty.)

3. Mechanism: the Sound Verifier-Frontier Ratchet

A slice is the set of facts a sound referee can certify. The ratchet, per rung:

1. **Fuel.** A different-execution-class sound oracle labels items the current model fails at high k (off-support, base-fail@k) — verified-correct trajectories the base cannot sample (SGDF).
2. **Fold.** Train new capacity (a frozen-base + new-LoRA CIP rung) on those labels; old capability is frozen (no forgetting), the new capability is permanent weights.
3. **Discover (next rung).** To reach the next capability, search over compositions of the certified library and admit one only if the referee certifies it (verified search, not learned routing).
4. **Grow the slice.** Author a new checker by composing certified checkers; admit only under the independent referee on a fresh adversarial battery. The slice grows.

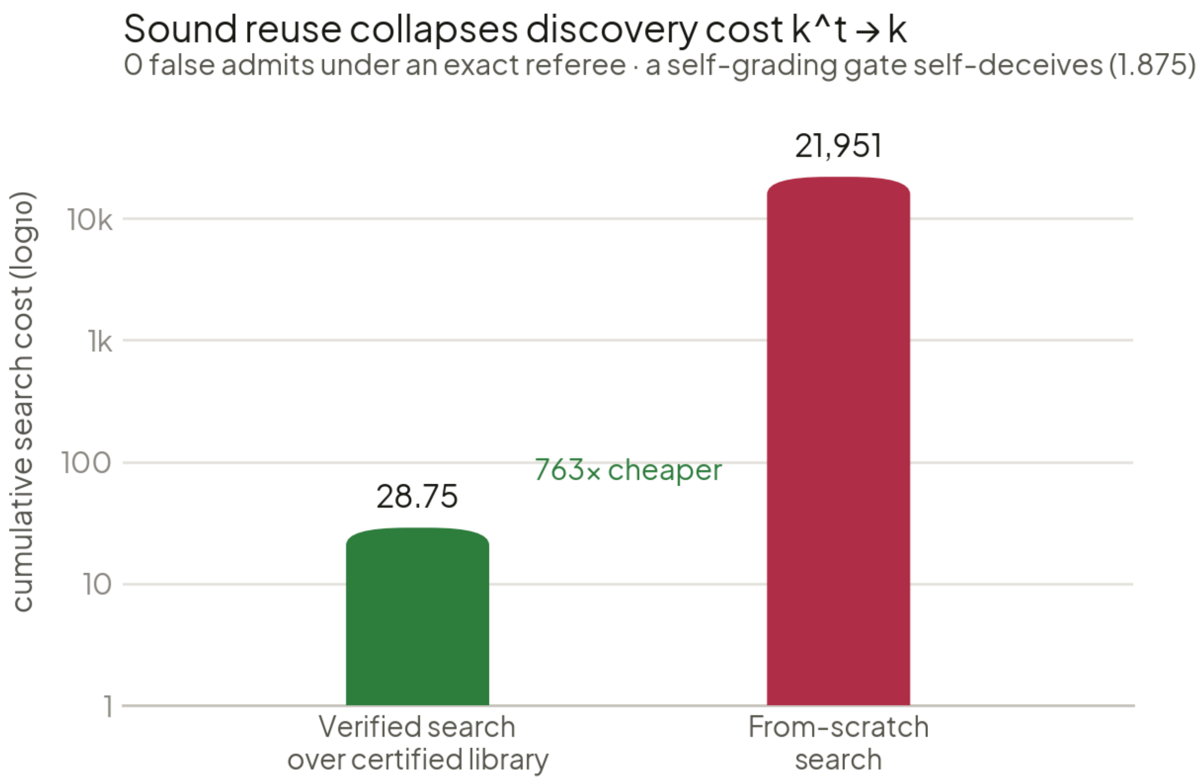
The **soundness invariant**: every admission is gated by a referee that (a) is a different execution class from the learner, (b) never reads test answers, (c) re-derives ground truth independently. The **diagnostic invariant**: track believed_frontier (what the system's own slice certifies) vs true_frontier (an independent held-out referee). Leakage signature = believed ≫ true.

4. Experiments (toy; pre-registered; matched controls; leakage-audited)

All code under `research/right_ai/sandbox/` (+ `verify_comp_*`). Oracles are exact/symbolic (Python permutation algebra / integer predicates), fixed power, never an LLM, never reading test answers; the model runs alone at test. Each result is a pre-registered PASS/KILL.

4.1 Naive reuse is killed. Reusing the model's own approximate learned modules accumulates error down the chain and still diminishes (own-chain slope −0.088 vs open-loop −0.130). Handing the model the exact decomposition makes the frontier flat (slope +0.001) — but a diagnostic shows this is built-in-the-answer (frontier == depth by construction), and when the model must discover the decomposition by routing, it collapses to open-loop (slope −0.088 ≈ −0.066). Verdict: result-holds-but-narrow; "non-diminishing via reuse" is false unless reuse is sound and discovery is solved.

4.2 C1 — discovery via verified search (PASS). `compounding_discovery_v2.py` (permutation group $\approx 3.6\times10^6$; 8 seeds). REUSE_SEARCH reaches all tiers at bounded cost (per-tier slope −0.08; cumulative 28.75) with 0 false admits; from-scratch search stalls and costs 763× more (cumulative 21,951); a weak/self-grading gate self-deceives (1.875 false admits). The discovery that collapses under gradient-routing works under verified search over a certified library: reuse collapses $k^t \to k$, and exact-referee-gating kills self-grading. Honest read: library-learning + verifier-as-filter in kind; the contribution is the bounded-cost + soundness separation.



SOURCE: sound-compounding-ratchet.md §4.2 (compounding_discovery_v2.py, 8 seeds).

FIGURE 2 Cumulative discovery cost, log scale: verified search over a certified library (28.75) vs. from-scratch search (21,951) — 763× cheaper — with 0 false admits under an exact referee (`compounding_discovery_v2.py`, 8 seeds).

4.3 C2 — sound slice-growth and its hard cap (PASS). `slice_closure_killtest.py` (8 seeds). A seed referee certifies only atomic facts; authored checkers compose it. Result: reach 30 compound facts beyond the atomic seed with 0 false admits, but a "rogue" function outside the seed's closure (a new execution class) is admitted soundly 0 times, while a self-grading gate admits all 20. The slice grows soundly to the closure of the seed execution class and is hard-capped there; a new execution class requires external off-support fuel, not self-authoring.

4.4 Slice-growth ratchet + the believed-vs-true test (PASS). `verify_comp_slicegrowth.py`. With a finite slice that must author checkers to extend, gated by an independent different-execution-class referee: **GROW_SOUND** true-frontier 6.75 vs frozen-slice 1.0, true_slope +0.83 (non-diminishing), 0 false admits, believed==true (calibrated); **SELF_GRADE / GROW_UNSOUND** believe 8.0 but truly 6.75 (gap +1.25, 5 false admits). The believed-vs-true gap is the self-improvement leakage made measurable; the referee rejects a vacuous always-accept predicate, so the lift is not a permissive gate.

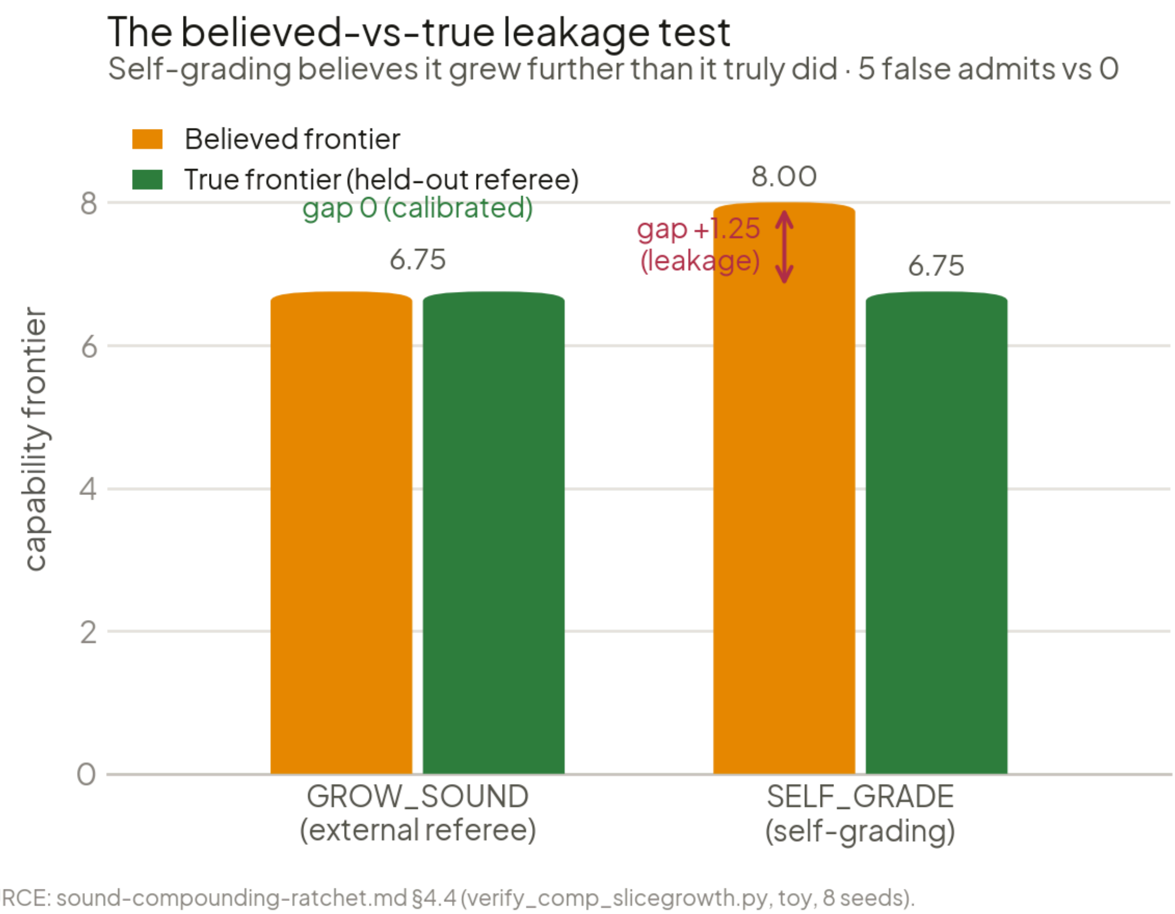


FIGURE 3 Believed vs. true frontier: GROW_SOUND believed=true 6.75 (gap 0, calibrated, 0 false admits) vs. SELF_GRADE believed 8.00 / true 6.75 (gap +1.25, 5 false admits) (`verify_comp_slicegrowth.py`, toy, 8 seeds).

5. Limitations and honest scope

1. **Toy scale.** Symbolic permutation/integer domains and tiny nets. Real LLMs add the deep-feature / position-entanglement obstacle (our prior RoPE blocker) and the open problem of authoring sound checkers at scale. No claim transfers to real models until §7 runs.
2. **The cap is real.** The ratchet is uncapped only within the seed execution class's closure. Genuinely new capability is fuel-gated — it requires an external sound oracle of a different execution class. We do not show open-ended capability from nothing; we show sound, no-forgetting accumulation given a fuel supply.
3. **Novelty is modest.** The pieces are known (library learning, verifier-as-filter, CIP, SGDF). The contributions are the composition, the soundness gate as the load-bearing element, and the believed-vs-true leakage test — not a new primitive.
4. **Discovery solved only via search.** Gradient-learned routing still collapses (§4.1); our positive discovery result is verified search, whose cost we bound on a toy but have not stress-tested at

library sizes where search itself could become the bottleneck.

6. The believed-vs-true test as a community tool

We propose `believed_frontier − true_frontier` (system's-own-slice certification minus an independent held-out referee) as a standard audit for any self-improvement / RSI claim: a loop that believes it ratcheted while its true frontier tracks a sound gate is self-grading, not improving. In our toy it cleanly separates sound growth (gap 0) from self-grading (gap +1.25). This is cheap and we recommend it as a required control.

7. Pre-registered real-LLM test (the decisive next experiment)

`research/right_ai/colab/real_llm_compounding_rung.py` (free Colab T4). Qwen2.5-0.5B, tiered cumulative modular-arithmetic chains, exact Python oracle. Establish the wall (base fail@8 on deep chains = off-support). **RATCHET** (carry the CIP-LoRA forward, SGDF-fuel each tier) vs **NO_FOLD** (fresh LoRA per tier), matched label budget, with a forgetting probe. **PASS:** RATCHET deep-tier pass@1 ≥ NO_FOLD + 0.15, forgetting < 0.05, fixed-power Python oracle. **KILL:** reuse doesn't help / no wall / lift vanishes with answer-only labels (leakage). This is the toy→real bridge; until it runs, every claim here is toy-scoped.

8. Conclusion

If scaling is a fixed diminishing curve, intelligence must be acquired and accumulated. We give toy-scale, pre-registered evidence that a verify-grow-reuse loop can compound soundly — sound discovery at bounded cost, sound slice-growth to the execution-class closure, no forgetting, and a measurable separation from self-grading — while being honest about the cap (new capability is fuel-gated) and the scope (toy until §7 runs). The believed-vs-true test is offered as a reusable guard against the self-deception that has dogged self-improvement research. The mechanism is a modest, honest recombination; its value is that every load-bearing claim has a passing kill-test and a pre-registered path to real scale.

Reproducibility: all experiments are deterministic, pre-registered, matched-controlled, and leakage-audited per the `deep-ai-ml-research` protocol; code + result JSONs are in the repo. Companion: `COUNCIL_scaling_laws_2026-06-29.md` (Layer 1), `COUNCIL_compounding_2026-06-29.md` (full council log).

VEXT LABS · VL-2026-05 Claims are scoped. Records are public.