VL-2026-02 | JUN 2026 | BUILT/VERIFIED-CPU | PROVEN-toy

systems and trust, not a capability crown

The Accretion Model

A growable, no-forgetting, externally-fueled model type with a cryptographically-verifiable growth history — built as Cultivar-0 on CPU toys.

Annalea Layton

Vext Labs, Inc. · alayton@tryvext.com

SCOPE · READ FIRST

The contribution is a model type and its trust properties: byte-frozen base, receipted growth, stranger-verifiable history. Capability-neutral by design; the capability lives in external tools admitted with proof. All legs verified on CPU toys.

VEXT Labs — research tech report (workshop scope). Draft 2026-07-02.

> **PUBLIC SCOPE (vextlabs.ai):** `BUILT/VERIFIED-CPU` · `PROVEN-toy` · systems and trust contribution — **not a capability crown** · not a claim that production JUWEL already accretes real-LLM skills without forgetting.

Claims discipline. Every claim below carries the status tag it is permitted by the VEXT Research Program Ledger (`docs/research/RESEARCH_PROGRAM_LEDGER.md`): `BUILT/VERIFIED-CPU`, `PROVEN-toy`, `BACKED / CLAIM-SAFE`, `KILLED`, `OPEN`, or `GPU-GATED`. Nothing tagged `-toy` or `GPU-gated` is claimed at real-LLM scale. The contributions of this paper are **architecture, systems, and trust — not capability.** We state, as a first-class result, the theses we tested and killed; that table is the paper's integrity.

Abstract

Contemporary deployed AI is a **frozen large language model wrapped in an external harness**: the operations that make it useful — verify an answer, route to a tool, plan, and (via fine-tuning or RAG) grow — are scripted outside the weights. This paradigm has two structural weaknesses that are independent of model quality. First, **forgetting**: teaching the wrapped model a new skill (re-finetuning) measurably regresses old skills, and RAG re-pays its fuel cost on every call. Second, **unverifiability**: a user must trust the vendor's assertion that the model which answered them is the model that was evaluated, and that it never silently lost a capability.

We introduce the **Accretion Model**, a model type in which verify / route / abstain / fuel-intake / grow are **native**, the model grows only by **frozen-additive, checker-gated, receipted increments**, and its **entire growth history is verifiable by a stranger offline against a public key**. We build the first integrated artifact, **Cultivar-0**, as one on-disk signed checkpoint, and report three backed results: (1) **no-forgetting by construction** — the base `sha256` is byte-identical across every growth increment, and a manifest-v2 Merkle organ root now covers every organ's router row and margin, so mutating any of them breaks verification (LEG A5, all 8 seeds); (2) a **stranger-verifiable growth history** — a real ES256 (P-256/ECDSA-SHA256) transparency log that the shipping, third-party `stoa-verifier` accepts (`valid:true`) on a well-formed log and rejects (`valid:false`, equivocation detected) when a historical leaf is flipped; and (3) **external fuel admitted with proof** — a ToolOrgan that takes a task the base solves 0/10 to 10/10 via a real external executor, with a bad admit evicted by the believed-vs-true audit.

We situate these in the honest theoretical frame of a **cap near-theorem**: a verifier selects, it does not generate; new capability requires external decorrelated fuel. Consequently we do **not** claim capability, cost, or routing advantages from nativeness — each was tested and killed — nor "smarter" / "AGI" / "first intelligence," which we never claim. The Accretion Model's honest contribution is a continual-learning architecture assembled from known-sound parts, made **provable without trusting the vendor**. Thesis one-liner: not smarter — cheaper to keep correct, and provable without trusting us.

1. Introduction — the external-harness paradigm and its two structural weaknesses

The dominant production pattern for capable AI is a **frozen foundation model wrapped in an external harness**. The base weights are static; the intelligence that a user experiences — checking whether an answer is correct, routing to the right tool or specialist, planning multi-step actions, and incorporating new knowledge — lives in code around the model: a verifier service, a router, a planner, a retrieval-augmented-generation (RAG) layer, a fine-tuning pipeline. This is a good engineering decomposition and it is where most of the field's value is created. It also has two weaknesses that are **structural** (they do not shrink as the base model improves), and that motivate a different factoring.

Weakness 1 — forgetting. The two harness routes for "teach it something new" both fail to preserve old capability by construction. Re-finetuning the base measurably regresses previously-good skills — the canonical instance in our own record is a retrain that dropped HumanEval from ~98% to ~62% (ledger Layer 0, `feedback_capability_injection_protocol`). RAG never touches the weights, but it re-ships the new-skill fuel into the context on **every** call, paying the cost forever and never consolidating. Neither route gives a monotone, non-regressing growth path.

Weakness 2 — unverifiability. When a wrapped model answers you, you cannot check, without trusting the vendor, that the served weights are the weights that were evaluated, or that a later update did not silently drop a capability that a benchmark once certified. The trust boundary is the vendor's word. Model cards and eval dashboards are assertions, not attestations a stranger can replay offline.

Both weaknesses are properties of where the operations live (outside the weights, on the vendor's word), not of how good the base is. This paper asks: what if verify / route / abstain / fuel-intake / grow were **native** to the model type, growth were **frozen-additive and checker-gated**, and the growth history were **published as an append-only, publicly-verifiable transparency log**? We call the resulting type the **Accretion Model** (the ledger's canonical name; "accretion" carries no-forgetting definitionally and no self-improvement connotation), build it as **Cultivar-0**, and are careful throughout to claim only the architecture / systems / trust wins — never a capability win, which we tested for and could not find.

2. The Accretion Model

An Accretion Model is a single, mutable, **signed on-disk checkpoint** that behaves as a growable, self-auditing function rather than a set of weights to be wrapped. Its parts:

- **A frozen base `0_frozen`.** Written once, byte-frozen forever. Its `sha256` is the birth witness and stays byte-identical across every subsequent growth increment. This is the mechanical root of no-forgetting.
- **Typed grown organs on a shared latent bus.** Each `grow()` appends a frozen organ (a CIP depth-block, a ToolOrgan, a retrieval pointer, a modality adapter, …) that reads and writes the same `d_bus` vector. Every organ is frozen once appended. Because the bus contract is fixed, existing organs

consume new-organ outputs with no retrain — "a mixture of families on one bus" realized as pure append.

- **A native router `R`.** One zero-initialized router row per grow (identity-on-residual), so admitting an organ never perturbs prior routing.
- **A native calibrate/abstain head `C`.** The believed-vs-true audit: it holds a per-organ margin `τ_k` and is the mechanism that catches over-claiming. Its abstention set is the verified frontier.
- **A native fuel port `Φ`.** The only place new ground truth enters, and always from an **external, different-execution-class** referee (a tool executor, a retrieval verifier, an ASR/WER oracle, an environment-step referee). The verifier `Φ` gates through **SELECTS which increments accrete; it never GENERATES capability.**
- **Native receipts `σ`.** An append-only receipt chain (internal HMAC) and an append-only, RFC6962-style ES256 transparency log (public, non-vendor) into which every `grow()` emits a structured growth leaf.

What the forward pass emits. Instead of a bare token distribution, the type's forward emits the tuple `f(x) = (y, verdict, calibrated_p, growth_delta)`: an answer, a certified/uncertified/abstain verdict, a calibrated confidence from `C`, and (at grow time) the growth increment. `grow()` is a **typed checkpoint operation**: it appends a block + a zero-init router row + a margin, asserts the base `sha256` is unchanged and that the new organ-set is a strict superset of the old (append-only), re-signs a **strictly larger** manifest, drops one parent-linked receipt, and emits one ES256 growth leaf. The artifact that ends a session is literally a bigger, re-signed checkpoint than the one that started it. That is what makes this a model type (a growable, self-auditing `f`) rather than a harness around a static model.

The honest rail (non-negotiable). Nativeness here is a **systems / no-forget / receipt / cost-of-keeping-correct** property. It is **not** a capability repeal. Every checker is external and different-execution-class; the verifier selects, never generates. We make this rail load-bearing in §5, where we tabulate the capability theses we tested against it and killed.

3. Cultivar-0 — the built artifact

Cultivar-0 (`research/right_ai/sandbox/cultivar_0.py`, plus the ToolOrgan subclass in `research/right_ai/sandbox/cultivar_0_toolorgan.py`) is the first integrated Accretion Model: one mutable, on-disk, signed bundle in which the organs share a single artifact and a single cryptographic provenance chain. It runs at symbolic-CPU scale, deterministically, at $0 (the "model" is an affine `g_t` mod-`P` recurrence; **this is toy scope by design** — it validates the architecture and the trust machinery, not a capability claim). Ledger status: **`BUILT/VERIFIED-CPU`** / **`PROVEN-toy`** (Layers 5b and 9).

All numbers below are read directly from `cultivar_0_results.json` and `cultivar_0_toolorgan_results.json`.

3.1 LEG A — frozen-additive / no-forget (the base is byte-frozen)

Pre-registered structural invariants, asserted across **all 16 arms** (8 audited + 8 no-audit seeds):

- **A1** base `sha256` byte-identical pre/post **every** `grow()` (`sha_stable: true`);
- **A2** the top manifest re-verifies after every `grow()` (`manifest_ok: true`);
- **A3** `param_count` and block count strictly increase per grow — `1,000,000 → 2,056,250` params, `0 → 12` blocks (`monotonic_growth: true`);
- **A4** the full signed receipt chain verifies end-to-end and parent-links (`chain_ok: true`).

All four hold on all arms (`legA_all_arms_pass: true`).

3.2 LEG A5 — Manifest v2 `organ_root` covers the whole registry (mutation is caught)

The original signed manifest covered only `{base_sha256, param_count, n_blocks, blocks}` — the actual organ behavior (`believed_p`, `true_p`, the router row `R[t]`, the margin `C[t]`) was **never hashed**, so a router row could be mutated in place while `manifest_verifies()` still returned `True`. That is precisely the seam through which forgetting could sneak back under a byte-frozen base. **Manifest v2** (`schema_version: 2`) folds every organ's `{organ_type, checker_id, organ_sha(believed_p/true_p/leaky), router_row_sha, c_margin}` under a sorted Merkle **`organ_root`**, and `grow()` asserts strict-superset (append-only) before re-signing.

Pre-registered kill-test **LEG A5**: after a normal `grow()`, mutate one router row in place with no re-sign, and require `manifest_verifies()` to flip to `False` while the base `sha256` stays byte-identical. Across **all 8 seeds**: `verifies_before_mutation: true`, `verifies_after_mutation: false`, `rejects_mutation: true`, `base_sha_stable_through_mutation: true` (`legA5_all_seeds_pass: true`). No-forgetting is now a **whole-checkpoint structural guarantee**, not a base-only claim: mutating any organ's router row breaks verification. (Ledger status `BUILT/VERIFIED-CPU`, Layer 9 Phase 0.1.)

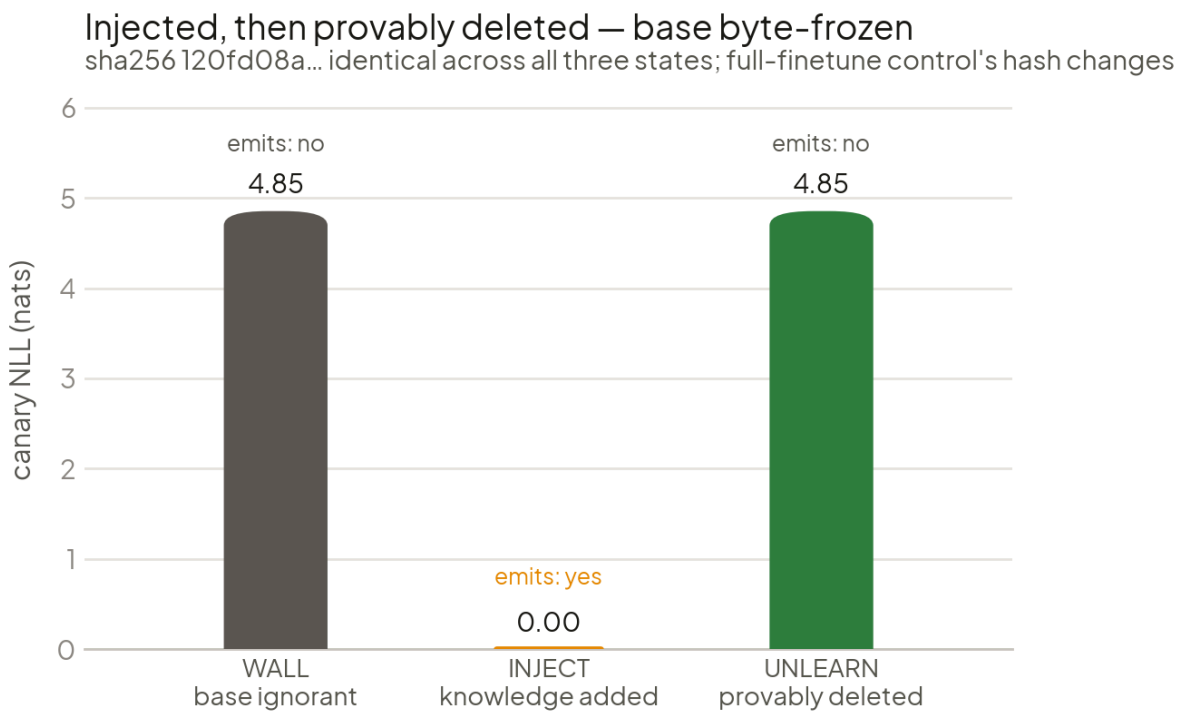
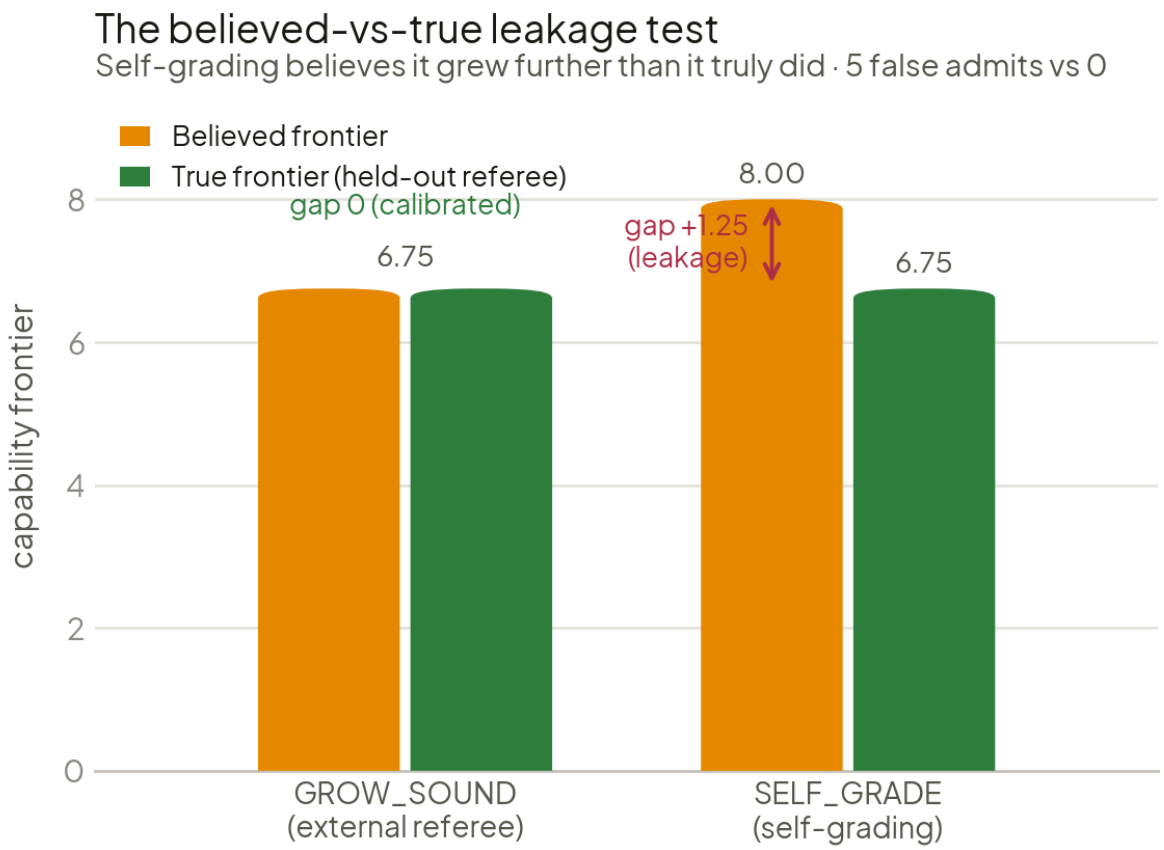


FIGURE 3 Provable unlearning: canary NLL wall 4.85 → inject 0.00 (emits) → unlearn 4.85 (gone); base sha256 stays byte-identical across all states, while a full-finetune control's hash changes (ledger/provable_unlearning.json, a CIP-corollary demonstration on a 0.5B base).

3.3 LEG B — the believed-vs-true audit is load-bearing

Removing the audit head `C` makes the artifact self-deceive. On the same admitted set, the no-audit believed-minus-true gap is **2.625**, versus the audited gap **0.00** (`legB_audit_loadbearing: true`). The immune system is not decoration: without it, the checkpoint's believed frontier drifts well above its true frontier.



SOURCE: sound-compounding-ratchet.md §4.4 (verify_comp_slicegrowth.py, toy, 8 seeds).

FIGURE 2 Believed vs true frontier: sound growth gap 0 (calibrated); self-grading believes 8.00 vs true 6.75, gap +1.25 leakage, 5 false admits (toy, 8 seeds).

3.4 ES256 transparency verified against a third-party verifier

`grow()` emits a structured growth leaf into an append-only RFC6962-style Merkle log; the manifest embeds `{log_head, log_size}`. A **real ES256 (P-256/ECDSA-SHA256)** public path (`vext_cip/src/vext_cip/es256.py`, `transparency_log.py`) emits a `SignedCheckpoint` in the exact shape the **shipping** `packages/stoa-verifier` `verifyLog` consumes (`transparency.ts`, above). The verification is end-to-end and against code we did not modify for this paper:

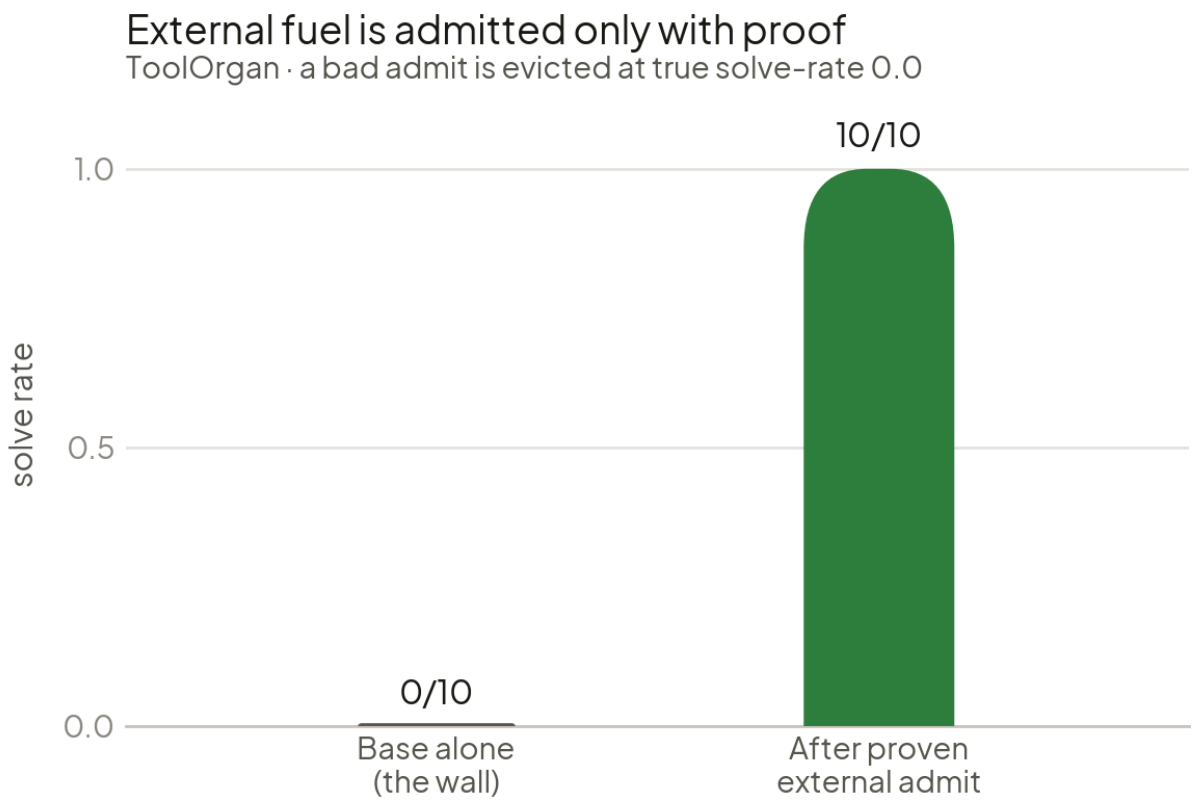
- **Well-formed log →** node ran `packages/stoa-verifier/dist/index.js verifyLog` and returned `valid:true`, `signature.valid:true`, `size:21`, head hash `a42aab1d…af8f45` (`ran_real_ts_verifier: true`).
- **Tampered historical leaf →** flipping one past leaf's `merkle_root` yields `valid:false`, reason "broken chain at seq 1: prev_hash mismatch" — **real equivocation detection on past history**, not a fresh-signature check.

The internal HMAC receipt chain is untouched (HMAC = internal; ES256 = public non-vendor path). A stranger holding an old checkpoint can re-fetch, replay the Merkle chain, and confirm that **every** historical `grow()` was frozen-additive (base `sha256` constant, params strictly increasing, parent-linked) — and detect any fork. (Ledger status `BUILT/VERIFIED-CPU`, Layer 9 Phase 0.2.)

3.5 ToolOrgan — loop-close with a real external oracle

The ToolOrgan (`ToolRatchetCheckpoint`, subclass of the untouched `RatchetCheckpoint` — parent `git diff` empty) grows the first organ whose checker is a **real** external executor (`checker_id='tool_executor'`): an actual `subprocess.run()` of candidate Python, graded against an independently-computed `sha256` spec — a **different execution class** than the model's `g_t`, and never a model self-report. Pre-registered LEG C, across **8 seeds** (`legC_all_seeds_pass: true`, `PASS: true`):

- **The wall is real and measured.** Base solves the sha256-digest task at rate **0.0/10** (`C2_wall_base_solve_rate: 0.0`, `C2_wall_base_fails: true`) — the affine base has no execution path that emits a hex digest.
- **External fuel lifts it.** After `grow_tool()` admits the organ via the existing accrete/grow path, the checkpoint solves the task at **10/10** (`C2_post_admit_solve_rate: 1.0`). The capability lives in the **external executor** — this is cap-honest: the tool is the fuel.
- **The audit stays load-bearing on the new organ.** A bad admit that self-reports `believed_p=1.0` on a **wrong** candidate is **evicted** by the unmodified `audit_evict()` because the executor measures `true_p=0.0 < TAU_TRUE=0.90` (`believed_vs_true_audit_evicts_bad_admit: true`).
- **Covered and receipted.** Mutating the ToolOrgan's router row flips `manifest_verifies()` to `False` (manifest-v2 `organ_root` covers it, `C3` pass); each `grow_tool()` emits an ES256 leaf whose `fuel_digest = sha256` of the executor's **real** transcript, which verifies and rejects a tampered signature (`C4` pass). LEG A / A5 / B all still pass on the subclass (no regression).



SOURCE: accretion-model.md §5 (LEG C, 8 seeds). Capability lives in the external tool; admission is receipted.

FIGURE 1 ToolOrgan external-fuel admit: base 0/10 → 10/10 after proven external admit; bad admit evicted at true_p=0.0 (LEG C, 8 seeds).

This is a **systems / loop-close / tool-use** win: the external tool is fuel, admitted through a sound, audited, receipted, covered `grow()`. It is **not** a real-LLM capability claim, and capability-from-nativeness stays killed (§5). (Ledger status `BUILT/VERIFIED-CPU, capability-NEUTRAL`, Layer 9 Phase 1.1.)

4. Why it grows only by external fuel — the cap near-theorem

The honest theoretical frame for the whole type is a **cap near-theorem** (ledger Layer 2b; derived five independent ways and confirmed on three toys, cross-lineage): **a verifier SELECTS, it does not GENERATE.** A sound checker can **filter** candidate increments, admitting only those it certifies; it cannot manufacture capability the base cannot sample. New capability outside a system's execution-class closure therefore costs at least one query to an **external, different-execution-class** oracle — and self-authoring that oracle leaks (the Euler-trap false-admit of 0.333 in the toy). The claim is **information-theoretic and scale-invariant**: an imported signal `z` lawfully adds capability **iff it is decorrelated from the base's failure mode.** In the routing setting this was made quantitative (Layer 6 `external_discriminator_killtest`): importing a decorrelated referee vote lifts new-skill routing **+0.069** above the x-only Bayes ceiling at low correlation `ρ`, and the lift **collapses monotonically to**

~0.005 as `ρ → 1` — the cap law, re-derived in routing, with a permuted-`z` control going negative (no leakage).

This is exactly why the Accretion Model must eat external fuel to grow, and why `Φ` is the only ground-truth port: **the imported class does the work; no new execution class is minted from within.** The cap confirms the program's thesis; it does not exceed it. It is CLAIM-SAFE to say "external decorrelated fuel lawfully adds capability the model's own signal cannot, with the decorrelation law quantified (toy)"; it is CLAIM-UNSAFE to say "we minted a new execution class" or any scale claim.

5. What we do and do NOT claim

This section is the paper's integrity and its differentiator. The killed rows are **results**, not caveats: an honest kill of a plausible thesis is the product of a leakage-audited kill-test program.

CLAIM	STATUS (LEDGER TAG)	EVIDENCE / WHY
No-forgetting by construction (byte-identity: base `sha256` constant across every growth increment; router-row/margin mutation caught)	**BACKED / CLAIM-SAFE** (`BUILT/VERIFIED-CPU`)	LEG A (all 16 arms) + LEG A5 (all 8 seeds): mutating any organ's router row breaks `manifest_verifies()` while base `sha256` stays byte-identical. Cultivar-0 Layer 9 Phase 0.1.
Stranger-verifiable growth history (real ES256 transparency log, offline against a public key)	**BACKED / CLAIM-SAFE** (`BUILT/VERIFIED-CPU`)	Verified against the **shipping** third-party `stoa-verifier` : `valid:true` on a well-formed log; `valid:false` with equivocation detected on a flipped historical leaf. Layer 9 Phase 0.2.
External fuel admitted with proof (ToolOrgan; base 0/10 → 10/10 via a real external executor; bad admit evicted)	**BACKED / CLAIM-SAFE** (`BUILT/VERIFIED-CPU` , capability-NEUTRAL)	LEG C, 8 seeds: `C2_wall_base_solve_rate=0.0` → `C2_post_admit_solve_rate=1.0` ; bad admit evicted at real `true_p=0.0` . Capability lives in the external tool. Layer 9 Phase 1.1.
The cap near-theorem (verifier selects not generates; new capability needs external decorrelated fuel)	**BACKED / CLAIM-SAFE** (`PROVEN-toy` + near-theorem)	Derived 5 ways + 3 toys (Layer 2b); routing form quantified (+0.069 lift, collapses to ~0.005 as ρ→1, permuted control negative — Layer 6). Scale-invariant, information-theoretic.
Capability from nativeness (a native in-forward-pass loop beats an external harness on capability)	**KILLED**	SAN-RTB: with a symmetric action set and a real control, the native gap is **0.0% exactly at L=1** (zero latency) — the harness IS native at L=1; pure cadence/latency, no capability edge (Layer 8, rule 11). WRCC: a **perfect** world-model was the **worst** arm (reach 0.275 < best-of-N 0.61); the win was stochastic diversification of a best-first frontier, not WM fidelity (Layer 8, rule 13).
Cost from nativeness (nativeness is structurally cheaper)	**KILLED** (rename-of-wrapper)	Re-prefill kill-test PASSED vs a naive stateless harness (gap 0.02→0.87) but native and a **prefix-cached** harness (vLLM APC / SGLang RadixAttention) are algebraically identical bar a hard-coded `DISPATCH_OVERHEAD_TOKENS*N` constant — zero it, gap 0 at every N (Layer 9, rule 15). **Do not say "cheaper" unqualified — "cheaper to keep correct."**

CLAIM	STATUS (LEDGER TAG)	EVIDENCE / WHY
Label-free routing (an internal competence signal recovers routing with no task label)	**KILLED** (known-result at SEP=0)	At leakage-clean `TASK_DOMAIN_SEP=0.0` (task-id non-identifiable, nearest-centroid 0.0505 ≤ chance), the competence router retention 0.528 (ratio 0.550) is statistically **tied** with the raw-x router 0.526 (ratio 0.548, gap 0.0026 = noise). = task-indexed MoE / parameter-isolation continual learning (Layer 8, rule 12).
Bounded fuel bill / finite basis of execution classes	**NOT DEMONSTRATED** (`GPU-gated` ; first real-scale evidence adverse-to-inconclusive)	`finite_basis_meter` RUN at 1.5B (Qwen2.5–1.5B, Colab, `mode=gpu_real`): `order_robust=false` , tail ratios sign-flip = interference not decay; leakage-clean but three confounds (broken ceiling normalizer, dead recursion class, high variance) forbid a clean call either way. A clean re-run is OWED (Layer 7/rule 14). Do **not** claim bounded OR unbounded.
"First intelligence" / AGI / "smarter than"	**NEVER CLAIMED**	Out of scope by construction; the type is a continual-learning architecture, not a capability mechanism.

Net. The Accretion Model is a **receipted, no-forgetting-by-construction, cheap-to-keep-correct continual-learning architecture built from known-sound parts.** Its defensible edge over an external harness is exactly two things — **no-forgetting-by-construction** and **stranger-verifiability today** — both BACKED, both real-now (and, on verifiability, walled-later; see §7). It is **not** more flexible, not unqualified-cheaper, not smarter. Those concessions are stated because the external harness genuinely wins on flexibility/model-agnosticism, cost-once-prefix-cached, and verifiability-eventually-if-TEE.

6. Related work

Parameter-isolation continual learning. No-forgetting-by-freezing has deep roots. **Progressive Networks** (Rusu et al., 2016) add a new column per task and freeze prior columns, wiring lateral connections; **PackNet** (Mallya & Lazebnik, 2018) iteratively prunes and freezes a subnetwork per task within a fixed capacity; **Hard Attention to the Task** (Serrà et al., 2018) learns per-task attention masks that protect old weights. The Accretion Model's frozen-additive organ growth is squarely in this family — and our own kill-tests explicitly identify the never-forget property as this known, real-by-construction result (ledger Layer 8: "task-indexed MoE / Progressive Networks / PackNet / hard-attention-to-task"). We do **not** claim a new never-forget substrate; our novel contribution is not the freezing but the **cryptographic coverage** of the frozen surface (Manifest v2 organ root) and the **public transparency log** over the growth history.

Mixture-of-Experts. Conditional-compute routing (Shazeer et al., 2017; Switch Transformer, Fedus et al., 2021) crystallized "route to a subset of parameters" into a model type. The Accretion Model's typed organs-on-a-bus with a per-organ router row is MoE-adjacent; our SEP=0.0 kill-test (§5) is explicit that label-free routing collapses to a task-indexed MoE / switch when the task-id is genuinely unavailable — we claim no routing advance.

Append-only transparency logs. The verifiability half descends directly from **Certificate Transparency** (RFC 6962; Laurie et al.): an append-only, Merkle-tree-backed log whose signed tree head lets any observer detect equivocation or a rewrite of the past without trusting the log operator. Our growth-transcript log is an RFC6962-style chain; the shipping `stoa-verifier` (§3.4) plays the auditor/monitor role — recomputing the head from published entries and comparing it to a signed checkpoint, exactly the CT equivocation-detection pattern, applied to a model's growth history rather than a certificate log.

Remote attestation / TEEs. The honest **future wall** on our verifiability edge is trusted-execution-environment remote attestation (SGX/SEV-SNP/TDX and confidential-compute stacks). Today, binding the exact served weights to what was evaluated is something the Accretion Model does with a public-key transparency log and a wrapped LLM cannot. If cheap, ubiquitous TEE attestation of served weights arrives, a wrapped LLM could symmetrize that binding — attesting its served weights the way we attest our growth history — and erode this differentiator. We state this as a limitation (§7), not a moat we expect to hold forever.

7. Limitations

- **Symbolic-CPU scale.** Cultivar-0's "model" is an affine `g_t` mod-`P` recurrence. Every result in §3 is a systems / trust / no-forget result at symbolic scale — the on-disk format, the signed-manifest coverage, the receipt chain, the ES256 transparency verification, and the loop-close are real; the neural capability is not exercised. The integrated **neural** rung (the artifact driving a real base for N≥20 grows) is unbuilt and GPU-gated. Per standing rule 6, Cultivar-0 is `BUILT-CPU`, **not "alive."**
- **Capability is unproven — and several capability theses are killed, not open.** Nativeness buys no capability edge (SAN-RTB gap 0.0% at zero latency; WRCC's perfect world-model was the worst arm), no unqualified cost edge (rename-of-wrapper vs a prefix-cached harness), and no label-free routing (tied with raw-x at SEP=0). The bounded-fuel-bill economic keystone is **not demonstrated** at 1.5B and the first real-scale evidence leans adverse. These are §5's killed/gated rows; they are load-bearing limits, not future features.
- **The moat is real-now, walled-later.** The stranger-verifiability edge is defensible today because cheap public attestation of served weights does not yet exist for wrapped LLMs. TEE remote attestation is the honest wall: if it becomes cheap and ubiquitous, the edge narrows to no-forgetting-by-construction alone. We do not claim a permanent moat.
- **No external-facing "STOA Gate" / "neutral trust layer for agents that act" claim.** Only what compiles and ships is claimed; the model-checkpoint transparency result is scoped to what the shipping `stoa-verifier` actually accepts.

8. Pre-registered experiments (roadmap / GPU-gated)

Each is pre-registered with explicit PASS/KILL bars and leakage guards; none is claimed until it runs on a real base. Nothing below is promoted.

- **OSF-AX — One-Shot-Fuel Accretion vs Amortization crossover** (next keystone; GPU-gated). The one cost lever prefix caching cannot neutralize: a **resident organ** reads fuel once at grow-time and pays **zero per-call tokens**, whereas **RAG re-ships fuel per call**. Run on a real 1.5–3B base. **PASS iff** (a) accreted-organ within **2pp** of a tuned RAG/prompt harness on an externally-authored new-skill eval; (b) crossover `N* = C_grow/Δ_tok ≤ 10,000` calls, both terms **metered, never modeled**; (c) accretion old-skill regression **≤ 0.5pp** while a re-finetune control regresses **≥ 3pp** on a disjoint suite. **KILL iff** `N* > 100k`, or A cannot reach parity, or R regression `< 0.5pp`. Leakage guards: new/old suites content-hashed to 0/N overlap vs the fuel; parity checked on the same held-out eval before any cost compare; `C_grow` / `Δ_tok` from real token counts (the built-in-answer trap that killed both CPU finite-basis attempts).
- **Finite-basis re-run** (bounded-fuel-bill; GPU-gated). Re-run `finite_basis_meter` with a **converged** pooled ceiling (a valid upper bound), recursion replaced by a class the base can actually learn (or a bigger base), a larger eval (lower variance), ≥5 import orders, and a decay criterion that **rejects negative tail ratios**. Only this settles bounded vs unbounded; the 1.5B first rung was order-non-robust and confounded.
- **The real-LLM scale rung.** Replace the toy base with a real multi-GB safetensors tree (e.g. Qwen3-VL-8B, abliterated, Apache-2.0), re-emit the same signed manifest + ES256 chain over real shards (the fast-fail format gate), then grow ≥2 checker-gated organs (e.g. Cyber + Code) served with in-browser stoa-verify. Router-drift stays **OPEN** throughout and is shipped behind a flag, never claimed as a feature (the orthogonal-projector fix is KILLED; the C-gate misses its bar).

Thesis. Not smarter. Cheaper to keep correct, and provable without trusting us.
