Beyond Human Comprehension: Designing Machine-Required Papers with Verifiable AI Expansion

Douglas C. Youvan

doug@youvan.com

www.youvan.ai

October 6, 2025

Some scholarly results cannot be fully grasped by unaided humans—not because they are poorly written, but because their *shortest faithful form* exceeds human working memory. We call these Machine-Required Papers (MRPs): articles whose core arguments or exhibits must be *expanded and narrated by an AI* to be understood. We formalize when "AI-required" is justified via three barriers—proof-length, representation, and compositional—and introduce a concrete design pattern: Machine-Required Capsules (MRCs) packaged with an AI Execution & Verification Protocol (AEVP). MRCs contain proof objects, vast search graphs, or high-dimensional structures; AEVP standardizes expansion, checkpointing, and *compute receipts* (model/version, seeds, walltime, step counts, hash chains). We specify acceptance criteria (necessity, verifiability, traceability, safety) so editors and reviewers can distinguish genuine beyond-human artifacts from rhetorical claims. Three worked blueprints—SAT/SMT certificates (MRP-DRAT-FACTOR), 8D topology tours (MRP-8D-TOPO), and million-step rewrite proofs (MRP-REWRITE-10^8)—show how to author MRPs while keeping the human narrative responsible for aims, assumptions, and ethics. The paper provides schemas, verifier cards, and a reproducible Repro Pack to make AI-required comprehension auditable, repeatable, and safe.

Keywords: machine-required paper, machine-required capsules, AI execution & verification protocol, compute receipts, proof replay, DRAT certificate, high-dimensional projections, persistent homology, exhaustive search, rewrite systems, acceptance criteria, traceability, preregistration, drift sentinels, governance, reproducibility.

A collaboration with GPT-5. CC 4.0. 61 pages.

Outline

1. Introduction
   1.1 Why “AI-required” scholarship, 1.2 Three barriers (proof-length, representation, compositional), 1.3 Contributions
2. Definitions & Problem Setting
   2.1 Human-checkable claims vs. machine-expanded substance, 2.2 Threat model (hallucination, drift, leakage)
3. Design Pattern: Machine-Required Capsules (MRCs)
   3.1 Capsule header (IDs, hashes, resources), 3.2 Inputs & dependencies, 3.3 Expected artifacts, 3.4 Verifiers
4. AEVP: AI Execution & Verification Protocol
   4.1 Verifier Cards (V-A…V-J: scope, proof replay, counterexample search, diagramming, units, receipts, transfer, glossary, citations, ethics)
   4.2 Checkpointing & hash chains, 4.3 Receipts schema
5. Acceptance Criteria for “AI-Required”
   5.1 Necessity, 5.2 Verifiability, 5.3 Traceability, 5.4 Safety & governance
6. Blueprint Exemplars
   6.1 MRP-DRAT-FACTOR (SAT/SMT certificate), 6.2 MRP-8D-TOPO (persistent homology tour), 6.3 MRP-REWRITE-10^8 (rewrite confluence/termination)
7. Authoring Workflow
   7.1 Human narrative responsibilities, 7.2 Capsule construction, 7.3 Preregistration & leakage guards
8. Evaluation & Governance
   8.1 Stability (mean±sd, ranking), 8.2 Drift sentinels, 8.3 Access control & dual-use scoping
9. Reader & Reviewer Guide
   9.1 Minimal steps to inspect, 9.2 Issue taxonomy, 9.3 Repro Pack contents
10. Risks, Ethics, and Limitations
11. Implementation Toolkit
    11.1 JSON schemas, 11.2 Minimal checker pseudo-code, 11.3 Reference receipts
12. Conclusion & Future Standards

References

1. Introduction

Modern scholarship increasingly intersects with results whose *shortest faithful form* cannot be fully inspected by an unaided human. This paper defines and operationalizes the Machine-Required Paper (MRP): a manuscript whose core argument or exhibit must be expanded and narrated by an AI—under a verifiable protocol—to be understood. We distinguish MRPs from conventional "AI-assisted" papers (where an AI is optional) by setting acceptance criteria (necessity, verifiability, traceability, safety) and by standardizing packaging via Machine-Required Capsules (MRCs) and an AI Execution & Verification Protocol (AEVP). The goal is not to excuse opacity; it is to discipline it, so that beyond-human content is auditable and reproducible.

1.1 Why "AI-required" scholarship

1. Reality of scale. Some valid arguments are *only* known through massive proofs/certificates (SAT/SMT DRAT logs, large-scale rewrite confluence proofs), exhaustive counterexample searches, or high-dimensional geometric analyses. For these, "make it shorter" is not feasible without losing correctness or scope.

2. Integrity over paraphrase. Human-readable summaries are necessary but insufficient: they can be eloquent and still wrong. An *AI-replayable* proof or search trace, coupled with receipts and checkpoints, lets readers *verify* rather than merely trust.

3. Standardization and fairness. Without a norm, authors can bury complexity off-paper or in ad-hoc code. MRPs force the substance into capsules with declared resources, hashes, and verifiers, giving editors and reviewers a clean contract for inspection.

4. Pedagogy with honesty. Students and reviewers can start with summaries, then request deeper layers (10 bullets → 1 page → transcript with checkpoints) while keeping pointers to the exact steps the AI replayed. This replaces "trust me" with *layered traceability*.

5. Safety and governance. When comprehension requires execution, we can enforce guardrails: preregistration (no post-hoc tuning), drift sentinels (model/version logging), and dual-use scoping (disable operationalization cards).

Bottom line: AI-required does not mean "mystical." It means the paper ships a *contract* for machine-mediated comprehension that anyone can expand and verify.

1.2 Three barriers to unaided human understanding

We formalize when “AI-required” is justified by identifying three independent barriers. An MRP needs at least one; many will have several.

(A) Proof-length barrier.

- *Definition.* The shortest known faithful argument requires ≥10^6–10^9 primitive steps (e.g., DRAT certificates, formal tactic scripts).
- *Symptom.* Humans cannot replay or even *sample* the argument without automation.
- *AI role.* Deterministic replay with hash-chained checkpoints and failure-on-mismatch. The AI produces layered summaries tied to checkpoint IDs.

(B) Representation barrier.

- *Definition.* The core object cannot be inspected directly because it resides in very high dimension or massive graphs (e.g., 8D persistent homology, million-node state spaces).
- *Symptom.* Any static 2D/3D figure collapses essential structure.
- *AI role.* On-demand projection and narration: generate labeled slices, tours, and quantitative summaries, each linked back to the underlying structure and checks.

(C) Compositional barrier.

- *Definition.* Correct understanding requires tracking thousands of interdependent invariants or constraints (non-local, non-linear), where local reasoning misleads.
- *Symptom.* Humans lose the global state; contradictions or missed edge cases proliferate.
- *AI role.* Maintain the global ledger/state, execute consistency checks, and answer scoped queries (e.g., “show the smallest violating configuration”) with proofs or counterexamples.

Each barrier is paired with verifier obligations: receipts (model/version, seeds, walltime, steps), deterministic checks, and a trace that binds summaries to machine-verified artifacts.

1.3 Contributions

1. Definition and acceptance criteria for MRPs.
   We formalize *when* it is legitimate to claim that AI is required:
   - Necessity: no substantially shorter faithful exposition exists (or the paper chooses the shortest known, still beyond human replay).

- Verifiability: third parties can expand the capsules and reproduce receipts.
- Traceability: every summary paragraph links to checkpointed artifacts (IDs/hashes).
- Safety: preregistration of constraints; drift sentinels; dual-use scoping.

2. Packaging standard: Machine-Required Capsules (MRCs).
   A schema for headers (IDs, hashes, dependencies, resource bounds), inputs (program/seed/constraints), expected artifacts (transcripts, projections, tables), and verifiers (checksums, spot-checks, property tests).
3. AEVP: AI Execution & Verification Protocol.
   A card-based protocol (V-A…V-J) covering scope mapping, proof replay with checkpointing, counterexample search, diagramming of high-D objects, units/sanity checks, compute receipts, transfer tests, glossary/citations, and ethics.
4. Blueprint exemplars.
   Three concrete patterns—MRP-DRAT-FACTOR, MRP-8D-TOPO, MRP-REWRITE-10^8—with capsule layouts, verifier checklists, and expected receipts, to guide authors in multiple domains.
5. Governance & evaluation toolkit.
   Stability metrics (mean±sd, ranking consistency), drift sentinels, preregistration templates, and minimal JSON schemas/checkers for editorial workflows.

Collectively, these contributions let authors publish beyond-human arguments responsibly, and let readers and reviewers expand, verify, and audit them with transparent, repeatable AI assistance.

2. Definitions & Problem Setting

This section fixes terms and the contract between the human-readable part of a Machine-Required Paper (MRP) and the machine-expanded substance that must be executed to achieve full comprehension.

2.1 Human-checkable claims vs. machine-expanded substance

Core definitions

- Human-checkable claim (HCC). A statement the paper expresses in plain language and (if needed) short math, such that a diligent reader can judge *scope, assumptions, and*

stakes without running any code or model.
Form: C := (premises P, guarantee G, scope S, limits L).

- Machine-expanded substance (MES). The *necessary* artifacts (proof objects, search traces, high-D structures, simulation transcripts) whose faithful inspection exceeds unaided human capacity and therefore must be expanded and narrated by an AI under a verifiable protocol.
  Form: MES := {capsules MRC_i, expansion tasks T_i, checks V_i}.
- Machine-Required Capsule (MRC). A signed, hash-addressed container with a header, inputs, resource bounds, and expected artifacts. It is inert text/binary until expanded.
  Header (minimal):
  {"claim_id": "...", "capsule_id":"...", "hash":"sha256:...", "deps":[...],
  "resources":{"steps_max":..., "ram_mb":..., "timeout_s":...}}
- AI Execution & Verification Protocol (AEVP). A card-based procedure that any conforming AI (or toolchain) must follow to expand MRCs, emit checkpoints and compute receipts, and bind natural-language summaries to verified artifacts.

Contract (who is responsible for what)

- Human narrative (authors) must:
  1. State HCCs with P/G/S/L in copy-safe prose/math.
  2. Justify *why* MES is necessary (which barrier(s): proof-length, representation, compositional).
  3. Provide the MRC headers, dependency graph, and evaluation rubric.
  4. Pre-register constraints and data partitions; disclose risks.
- Machine (conforming AEVP runner) must:
  1. Expand each MRC_i via Verifier Cards (V-A…V-J).
  2. Produce receipts: model/version, seeds, walltime, step counts, memory peak, hash chain.
  3. Emit checkpointed transcripts (e.g., every 10^4 steps) and projection specs for high-D objects.
  4. Bind every paragraph of its explanation to ⟨capsule_id, checkpoint_id⟩.
- Reviewers/readers can:

1. Inspect HCCs without a machine (understand *what* is claimed).
2. Run AEVP to verify *why* it holds (or fails), reproducing receipts and checkpoints.
3. File precise issues by capsule/checkpoint ("V-B-7a#chk=42 mismatch").

Trust boundaries and invariants

- Trust HCC prose for intent and scope, never for exhaustive detail.
- Trust MES only after AEVP expansion yields matching hashes/receipts.
- Invariant I1 (Binding). Every MES-derived sentence cites capsule_id + checkpoint_id.
- Invariant I2 (Determinism-to-receipt). Given identical MRC + seeds + version, receipts (hash chain, step counts) are reproducible up to allowed nondeterminism bands (declared in header).
- Invariant I3 (No silent browsing). Unless explicitly permitted, expansion uses only paper-provided inputs.

Examples (what is HCC vs. MES)

- HCC: "Under constraints R, property Q holds for all instances up to N."
  MES: DRAT proof object of unsatisfiability; replay transcript with checkpoints.
- HCC: "The dataset exhibits a persistent 8D cycle linked to variable V."
  MES: Simplicial complex + persistence barcode; projection tour script; numeric summaries.

2.2 Threat model (hallucination, drift, leakage)

We enumerate risks that can invalidate "AI-required" comprehension, and pair each with AEVP mitigations.

A. Hallucination & misbinding

- Threat. The AI fabricates steps, misquotes, or narrates content not derived from the capsule; or it cites the wrong checkpoint ("explanation drift").
- Impact. False confidence; unverifiable claims.
- Mitigations.
  - V-B (Proof Replay) hard binding: require per-step hash-chain; any gap → FAIL with minimal counterexample.

- Quote-by-location: every datum must cite ⟨capsule_id, checkpoint_id⟩ or section:line for HCC text.
- Uncertainty rule: on missing artifacts, output “uncertain” + list required inputs.

B. Model drift

- Threat. Provider updates change outputs across time for identical inputs.
- Impact. Irreproducible receipts; shifting narratives.
- Mitigations.
  - Receipts: log model, version, date, hash_chain.
  - Drift sentinels: re-run a tiny canonical subset weekly; publish deltas.
  - Pinning: prefer frozen endpoints for archival runs.

C. Data leakage / post-hoc fitting

- Threat. Ledger/constraints tuned using held-out data; prompts seeded with forbidden summaries.
- Impact. Inflated claims; invalid generalization.
- Mitigations.
  - Preregistration: commit to constraints, partitions, and metrics before expansion.
  - AEVP sandbox: no access to held-out data unless a card explicitly authorizes it; all I/O logged in receipts.
  - Red-team card: optional V-C’ that attempts to detect leakage patterns (e.g., suspiciously perfect fits).

D. Capsule tampering

- Threat. MRC bytes altered after review; dependency injection.
- Impact. Non-reproducible expansions; potential exploits.
- Mitigations.
  - Hashes & signatures: MRCs are content-addressed; signatures verified before expansion.
  - Dependency lockfile: resolve deps to exact hashes; fail on mismatch.

- Air-gapped verification: offline replay for security-sensitive capsules.

E. Nondeterminism & resource exhaustion

- Threat. Randomness or scheduling differences change outputs; runs time out.
- Impact. Flaky receipts; incomplete transcripts.
- Mitigations.
  - Seed discipline: fixed seeds in receipts; variance bands declared.
  - Bounded resources: headers specify steps_max, ram_mb, timeout_s; partial receipts must mark truncation.
  - Stability metrics: report mean±sd and ranking consistency across K jittered runs.

F. Dual-use and unsafe operationalization

- Threat. Expanded artifacts enable harmful replication or misuse.
- Impact. Real-world risk.
- Mitigations.
  - Scope cards: disable or redact operationalization steps; publish summaries only.
  - Access control: gated capsules; reviewer-only materials with audit logs.
  - Ethics card (V-J): requires explicit misuse analysis and mitigations.

G. Privacy & proprietary content

- Threat. Hidden data inside capsules; inadvertent disclosure during expansion.
- Impact. Legal/ethical violations.
- Mitigations.
  - Capsule hygiene: no personal data; synthetic or licensed assets only.
  - PII scanners & attestations: pre-expansion checks; receipts include “privacy_ok: true”.

Threat-to-control matrix (at a glance)

- Hallucination → hash-chained replay, quote-by-location, uncertainty rule.
- Drift → versioned receipts, drift sentinels, endpoint pinning.

- Leakage → preregistration, sandboxed I/O, red-team card.
- Tampering → content hashes, signatures, dependency lockfile.
- Nondeterminism → seeds, variance bands, stability metrics.
- Dual-use → scope cards, access control, V-J ethics.
- Privacy → capsule hygiene, PII checks, receipts attestation.

Together, these definitions and controls set the problem frame: MRPs separate *what must be human-legible* (claims, assumptions, ethics) from *what must be machine-expanded* (substance), and they make machine-mediated comprehension auditable, reproducible, and safe.

3. Design Pattern: Machine-Required Capsules (MRCs)

An MRC is a signed, content-addressed container that packages the *machine-expanded substance* of a result. It is inert until expanded by a conforming AEVP runner, which emits checkpointed transcripts, projections, and compute receipts. This section defines the on-paper contract.

3.1 Capsule header (IDs, hashes, resources)

Purpose. Identify the capsule unambiguously, bound it to a claim, declare resources, and freeze dependencies.

Header schema (copy-safe JSON):

```
{
  "claim_id": "C1",
  "capsule_id": "MRC-7a",
  "version": "1.0.0",
  "kind": "proof|search|projection|audit|simulation|other",
  "hash": "sha256:ab17...e2",
  "size_bytes": 3145728,
  "deps": [
    {"capsule_id": "MRC-6f", "hash": "sha256:9c2...77"}
```

```
 ],
 "resources": {
  "steps_max": 1000000,
  "ram_mb": 2048,
  "timeout_s": 900,
  "threads_max": 4
 },
 "nondeterminism": {
  "seed": 1729,
  "variance_bands": {"metric":"steps","sd_max":0}
 },
 "provenance": {
  "authors": ["A. Author", "B. Author"],
  "created": "2025-10-05",
  "license": "CC BY 4.0"
 }
}
```

Header invariants.

- Content addressability. hash is over the *capsule payload*; any byte change → new hash.
- Dependency lock. Each dep includes its exact hash; missing/mismatched → expansion FAIL.
- Resource honesty. resources are upper bounds; expansion must not exceed them without explicit “truncated” status in receipts.

3.2 Inputs & dependencies

Inputs. Structured parameters the runner may read at expansion time; everything else is considered *out of scope* unless permitted by a card.

Inputs schema:

```
{
  "params": {
    "dataset_ref": "doi:10.5281/zenodo.12345",
    "split": "heldout_v1",
    "alpha": 0.8,
    "bounds": {"N": 100000, "L": 100000}
  },
  "constraints": {
    "ledger": "sha256:3fa...1c",
    "prereg_id": "osf.io/abcd1"
  },
  "io_policy": {
    "allow_browse": false,
    "allow_net": false,
    "allow_files": ["./capsules/MRC-6f.bin"]
  }
}
```

Dependencies. Other capsules or fixed assets needed to expand this one (e.g., a search graph MRC required by a proof MRC).

Lifecycle (author view).

1. Draft capsule → compute hash → insert into manuscript header.
2. Freeze deps (hashes) → publish lockfile.
3. Upload inputs (e.g., ledger JSON) to a durable store → reference by content hash/DOI.

Trust boundary. If allow_browse=false, any outbound request by the runner is a protocol violation; receipts must log I/O.

3.3 Expected artifacts

Define *exactly what* expansion must emit, at what granularity. This prevents “hand-wavy” expansions.

Artifacts manifest (examples by kind):

- kind = "proof"
    - transcript.ndjson: line-delimited steps {idx, rule, goal_hash, ctx_hash}
    - checkpoints/ every 1e4 steps: chk_0001.json, … each with cumulative sha_chain
    - summary.txt: layered narrative (10 bullets → 1 page), with checkpoint_ids per paragraph
- kind = "search"
    - coverage.json: explored states, branching factors, pruned nodes (counts)
    - witnesses.json: smallest counterexample(s) or “none up to L”
    - profile.json: walltime by phase
- **kind = "projection"` (high-D → narrated tour)
    - slices/: specs of 2D/3D slices {axes, ranges, labels}
    - tour.md: ordered steps with captions and linked slice_ids
    - metrics.json: quantitative invariants (e.g., Betti numbers)
- kind = "audit"
    - issues.csv: rows {capsule_id, checkpoint_id, code, message}
    - repro_cmd.txt: exact invocation strings and seeds

Artifacts schema stub:

```
{
  "artifacts": [
    {"name":"transcript.ndjson", "role":"primary", "required":true},
    {"name":"checkpoints/", "role":"integrity", "required":true, "freq":10000},
    {"name":"summary.txt", "role":"narrative", "required":true, "layers":[10,200,1500]},
    {"name":"receipts.json", "role":"attestation", "required":true}
```

]

}

Binding rule. Every paragraph in summary.txt cites {capsule_id, checkpoint_id}; projections cite {slice_id} plus the source computation step.

3.4 Verifiers

Verifiers make the expansion *checkable* by third parties. They run during AEVP and are summarized in receipts.

Verifier Cards (AEVP subset).

- V-B Proof Replay. Deterministic re-execution; recompute sha_chain; abort on mismatch.
- V-C Counterexample Search. Exhaustive or bounded search; record coverage; output minimal witness if found.
- V-D Diagrammer. Validate projection specs, ensure labels match glossary, and that the DAG of relationships matches the manuscript.
- V-E Units & Sanity. Dimensional checks; order-of-magnitude tests.
- V-F Receipts. Emit:
- {
- "model":"GPT-5 Thinking",
- "version":"2025-10-06",
- "seeds":[42,1729],
- "walltime_s":612,
- "steps":1002318,
- "mem_mb_peak":1430,
- "sha_chain":["h0","h1", "...", "h100"],
- "status":"verified|failed|truncated",
- "io_log":[{"path":"./capsules/MRC-6f.bin","sha":"sha256:9c2...77"}]
- }

- V-G Transfer. Apply the same structure to a toy instance; confirm qualitative behavior.
- V-I Citations. Map narrative claims ↔ checkpoints; ensure one-to-one or one-to-few binding.
- V-J Ethics. Confirm preregistration IDs, leakage guards, and access scope.

Minimal verifier spec (per capsule):

```
{
  "verifiers": [
    {"card":"V-B", "params":{"checkpoint_freq":10000}},
    {"card":"V-F", "params":{"receipts_required":true}},
    {"card":"V-I", "params":{"bind_summary_to_checkpoints":true}}
  ],
  "failure_policy": {
    "on_checksum_mismatch":"fail-fast",
    "on_timeout":"emit_truncated_receipt",
    "on_missing_dep":"abort_with_reason"
  }
}
```

Pass/fail semantics.

- PASS (verified). All required cards succeed; status="verified"; hashes match; resources within bounds.
- TRUNCATED. Time/steps exceeded; partial artifacts + explicit truncation markers. Not acceptable for acceptance unless venue allows.
- FAIL. Any integrity mismatch (hash, binding, dependency) or verifier error.

Reviewer checklist (1 page).

1. Recompute capsule hash; verify against header.
2. Expand under AEVP; inspect receipts.json for model/version/seeds/sha_chain.

3. Open summary.txt; spot-check paragraph→checkpoint_id bindings against transcript.ndjson.
4. For projections, open tour.md; confirm cited slice_ids exist and correspond to emitted metrics.
5. Record stability: repeat K=5 with prompt jitter; report mean±sd; check ranking consistency.

Takeaway. MRCs turn “trust me” into a contract: a fixed header, sealed inputs and deps, explicit artifacts, and verifiers that any conforming runner can execute. This is the minimal structure needed to claim that full comprehension is machine-required yet auditable.

4. AEVP: AI Execution & Verification Protocol

AEVP is a card-based contract any conforming runner follows to expand Machine-Required Capsules (MRCs), verify integrity, and emit checkpointed artifacts plus compute receipts. It is model/tool–agnostic; everything needed to audit a run appears in the receipts and artifacts.

4.1 Verifier Cards (V-A…V-J)

Each card defines purpose, inputs, outputs, and failure modes. Cards may be required/optional per capsule.

V-A — Scope Map

- Purpose: Build a 7-bullet map of the claim family and the capsule graph.
- Inputs: Manuscript sections; capsule headers (deps).
- Outputs: scope.json with nodes (claims, capsules) and edges (depends_on, supports).
- Fail: Missing capsule or dangling deps → scope.status="incomplete".

V-B — Proof Replay

- Purpose: Deterministically replay a proof/search trace; recompute the hash chain.
- Inputs: Proof/search capsule payload; checkpoint_freq (e.g., 10^4 steps).
- Outputs: transcript.ndjson, checkpoints/, sha_chain[], replay_report.txt.

- Fail: Any step mismatch, checksum error, or non-monotone chain → status="failed", emit minimal counterexample.

V-C — Counterexample Search

- Purpose: Find a witness falsifying the property up to bound L, or certify “none up to L”.
- Inputs: Property spec, bound L, seeds.
- Outputs: coverage.json (visited states), witnesses.json (smallest witness or null).
- Fail: Coverage < declared lower bound; inconsistent witness; timeouts → status="truncated|failed".

V-D — Diagrammer (High-D Projections)

- Purpose: Generate labeled slice specs and a narrated tour of high-dim objects.
- Inputs: Projection capsule; glossary and label rules.
- Outputs: slices/*.json, tour.md, metrics.json (e.g., Betti numbers).
- Fail: Label mismatch with glossary; non-reproducible slice IDs; dangling references.

V-E — Units & Sanity

- Purpose: Dimensional and order-of-magnitude checks on reported quantities.
- Inputs: Manuscript equations; unit declarations; datasets/ledgers (by hash).
- Outputs: units_report.md with passes, violations, and suggested fixes.
- Fail: Unit inconsistency or physically impossible ranges → flag and fail if required.

V-F — Compute Receipts

- Purpose: Record attestation of the run.
- Inputs: Runner metadata; resource monitors; hash chain.
- Outputs: receipts.json (schema in §4.3), io_log.json (paths + hashes).
- Fail: Missing required fields; non-matching capsule hash; resource overrun without truncated.

V-G — Transfer Test

- Purpose: Apply the expanded structure to a toy instance; check qualitative predictions.
- Inputs: Toy spec; allowed I/O policy.

- Outputs: transfer_table.csv, transfer_summary.md.
- Fail: Toy outside declared scope; missing constraints; divergence from expected direction.

V-H — Glossary

- Purpose: Canonicalize terms/symbols as used *in this paper*.
- Inputs: Manuscript sections.
- Outputs: glossary.json (term, definition, first_seen, anchors).
- Fail: Collisions (same term, different defs) without disambiguation.

V-I — Citations & Binding

- Purpose: Bind narrative claims to checkpoints and to paper references.
- Inputs: summary.txt, transcript.ndjson, bibliographic metadata.
- Outputs: bindings.csv with rows {claim_id, capsule_id, checkpoint_id, ref_id, page_line}.
- Fail: Unbound summary paragraph; invented references; inconsistent anchors.

V-J — Ethics & Governance

- Purpose: Enforce preregistration, leakage guards, access scope, and dual-use redactions.
- Inputs: Prereg IDs; I/O policy; redaction policy.
- Outputs: ethics_report.md (risks, mitigations, compliance checklist).
- Fail: Violation of prereg or I/O policy; unsafe operationalization; absent risk assessment.

4.2 Checkpointing & Hash Chains

Goal: Make long expansions auditable via incremental integrity proofs and resumability.

Checkpointing rules

- Frequency: Declare checkpoint_freq in the capsule or receipts (e.g., every 10^4 steps).
- Contents (per checkpoint):
- {
- "checkpoint_id": 42,

- "step_idx": 420000,
- "state_hash": "sha256:…",
- "sha_chain": "sha256:…", # cumulative
- "walltime_s": 301.2,
- "mem_mb": 1184,
- "notes": "tactic: rewrite; goals=3"
- }
- Determinism: Given capsule bytes + seeds + version, recomputation of checkpoint hashes must be bit-exact (or fall within declared variance bands with a justification field).
- Resume semantics: A runner may resume from the latest verified checkpoint only if the receipts capture resume metadata (prior sha_chain_tail, walltime accumulators).

Hash chain construction

- Per step hash: h_step = H(rule || goal_hash || ctx_hash || metadata)
- Checkpoint hash: h_chk = H(h_chk_prev || concat(h_step[i…i+freq-1]))
- Global chain: sha_chain[k] = H(sha_chain[k-1] || h_chk), with sha_chain[0] = H(capsule_hash || seed || version)

Fail-fast condition: Any mismatch at step or checkpoint invalidates downstream hashes; the runner must halt and emit a minimal counterexample (the offending step, inputs, and expected vs. actual hashes).

Security notes:

- Use domain-separated hashing (H("AEVP/step" || …)) to avoid cross-protocol collisions.
- Store hashes as lowercase hex; include algorithm label (sha256:).
- For privacy-sensitive capsules, allow an air-gapped mode where only hashes/metrics (not raw states) are stored.

4.3 Receipts Schema

Receipts are the attestation record of an expansion. They must be machine-readable, self-contained, and verifiable.

JSON schema (copy-safe)


```
{
  "capsule": {
    "capsule_id": "MRC-7a",
    "hash": "sha256:ab17...e2",
    "kind": "proof",
    "version": "1.0.0",
    "deps": [{"capsule_id":"MRC-6f","hash":"sha256:9c2...77"}]
  },
  "runner": {
    "model": "GPT-5 Thinking",
    "version": "2025-10-06",
    "framework": "aevo:1.2.3",
    "hardware": {"cpu":"x86_64","ram_mb":16384,"gpu":"none"}
  },
  "execution": {
    "start_utc": "2025-10-06T14:12:03Z",
    "end_utc": "2025-10-06T14:22:15Z",
    "walltime_s": 612.2,
    "steps": 1002318,
    "mem_mb_peak": 1430,
    "threads": 4,
    "checkpoint_freq": 10000
```

```
},
"integrity": {
  "seed": 1729,
  "sha_chain": ["h0","h1","...","h100"],
  "determinism": "deterministic",
  "variance_bands": null
},
"io_log": [
  {"path":"./capsules/MRC-6f.bin","sha":"sha256:9c2...77","mode":"read"},
  {"path":"./artifacts/checkpoints/chk_0042.json","sha":"sha256:4a1...bb","mode":"write"}
],
"artifacts": {
  "primary": ["transcript.ndjson"],
  "integrity": ["checkpoints/"],
  "narrative": ["summary.txt"],
  "reports": ["replay_report.txt","units_report.md","ethics_report.md"]
},
"status": {
  "state": "verified | failed | truncated",
  "reason": null,
  "first_failure_checkpoint": null
},
"governance": {
  "prereg_id": "osf.io/abcd1",
  "privacy_ok": true,
  "allow_browse": false,
```

"allow_net": false

}

}

Validation checklist (editor/reviewer)

1. Capsule match: capsule.hash equals on-disk bytes; deps hashes resolve exactly.
2. Chain integrity: Recompute sha_chain from h0; confirm equality end-to-end.
3. Resource bounds: steps, mem_mb_peak, walltime_s ≤ header resources or marked truncated.
4. I/O policy: io_log contains only permitted paths; network flags false unless authorized.
5. Bindings: Open summary.txt; each paragraph carries {capsule_id, checkpoint_id} present in checkpoints/.
6. Repro spot-check: Re-run from checkpoint_id = k and confirm subsequent hashes.

Minimal example (abridged)

{

"capsule":{"capsule_id":"MRC-7a","hash":"sha256:ab17…e2","kind":"proof","version":"1.0.0","deps":[]},

"runner":{"model":"GPT-5 Thinking","version":"2025-10-06","framework":"aevo:1.2.3","hardware":{"cpu":"x86_64","ram_mb":16384,"gpu":"none"}},

"execution":{"start_utc":"2025-10-06T14:12:03Z","end_utc":"2025-10-06T14:22:15Z","walltime_s":612.2,"steps":1002318,"mem_mb_peak":1430,"threads":4,"checkpoint_freq":10000},

"integrity":{"seed":1729,"sha_chain":["h0","h1","…","h100"],"determinism":"deterministic","variance_bands":null},

"io_log":[{"path":"./capsules/MRC-7a.bin","sha":"sha256:ab17…e2","mode":"read"}],

"artifacts":{"primary":["transcript.ndjson"],"integrity":["checkpoints/"],"narrative":["summary.txt"],"reports":["replay_report.txt"]},

"status":{"state":"verified","reason":null,"first_failure_checkpoint":null},

"governance":{"prereg_id":"osf.io/abcd1","privacy_ok":true,"allow_browse":false,"allow_net":false}

}

Takeaway: AEVP operationalizes "AI-required" by prescribing *how* long arguments are expanded and *how* integrity is proven: via deterministic replay, hash-chained checkpoints, and machine-readable receipts that any third party can validate.

5. Acceptance Criteria for "AI-Required"

Not every hard paper is "AI-required." An AI-required claim is acceptable only if the manuscript meets four criteria—Necessity, Verifiability, Traceability, Safety & Governance—each with concrete tests, evidence, and failure modes.

5.1 Necessity

Definition. The core argument/exhibit cannot be faithfully conveyed in a human-length artifact without loss of correctness or scope; at least one barrier applies (proof-length, representation, compositional).

Tests (all required):

1. Barrier declaration: The paper names the barrier(s) and justifies them in one paragraph each.
2. Shortest-known form: Authors either (a) show a lower bound (e.g., minimal DRAT size, step count), or (b) adopt the *shortest known faithful* representation and state why any shorter would be unsound/incomplete.
3. Human-readable claim: The HCC (premises/guarantee/scope/limits) is understandable without the machine.

Evidence to include:

- Capsule headers with steps_max, size_bytes, dims/graph_size.
- A comparison table: "human synopsis length vs. capsule replay length."

Fail if: A reasonable human-length proof or figure exists and is omitted; barriers are asserted but unsubstantiated.

5.2 Verifiability

Definition. A third party can expand the capsules and independently check integrity *without* privileged access.

Tests (all required):

1. Reproducible expansion: Anyone with the paper and capsules can run AEVP and obtain receipts (model/version, seeds, walltime, steps, hash chain).
2. Deterministic replay or bounded variance: Given seeds and version, checkpoints reproduce exactly or within declared bands.
3. Integrity proofs: Long runs emit checkpointed hash chains; any mismatch halts with a minimal counterexample.

Evidence to include:

- Receipts schema populated in an example run (may be on a small slice).
- Dependency lockfile with exact hashes for deps.
- "How to run" one-page appendix (no external browsing unless whitelisted).

Fail if: Expansions need private keys/data; dependency hashes are absent; checkpoints cannot be recomputed.

5.3 Traceability

Definition. Natural-language summaries, numbers, and diagrams are bound to machine-verified artifacts at resolvable granularities.

Tests (all required):

1. Paragraph→Checkpoint binding: Every MES-derived paragraph cites ⟨capsule_id, checkpoint_id⟩.
2. Figure→Slice binding: Each diagram caption lists slice_id/projection spec and source steps.
3. Claim→Proof binding: A bindings.csv maps {claim_id → capsule_id → checkpoint_id → ref_id:page_line}.
4. Auditable deltas: If a capsule is revised, the paper increments version, updates hash, and includes a short "delta" note.

Evidence to include:

- summary.txt with inline checkpoint tags.
- bindings.csv in the artifact bundle.
- Glossary with canonical labels used across cards and diagrams.

Fail if: Summaries lack pointers; diagrams are hand-drawn without specs; numbers cannot be located in the transcript.

5.4 Safety & Governance

Definition. The manuscript constrains execution and disclosure so that machine-mediated comprehension is responsible and compliant.

Tests (all required):

1. Preregistration & partitions: Ledger/constraints, data splits, and metrics are preregistered (IDs in the paper).
2. I/O policy & sandbox: The capsule declares allow_browse/allow_net; receipts log *all* I/O paths and hashes; off-policy access is a fail.
3. Stability reporting: For any scalar results, report mean±sd across K jittered runs and ranking consistency (e.g., Kendall's τ).
4. Dual-use review: V-J ethics card addresses misuse risks; operationalization steps are redacted or gated when necessary.
5. Privacy & licensing: Capsules are free of PII and include data licenses; receipts include privacy_ok: true.

Evidence to include:

- Prereg ID (e.g., OSF/registry), I/O policy JSON, stability table, ethics report, license statements.
- Drift sentinel plan: small, periodic re-expansion to detect model updates.

Fail if: Post-hoc tuning on held-out data; network access without disclosure; no stability numbers; unsafe release of operational details.

Editor/Reviewer One-Page Checklist

- Necessity: Barrier(s) justified? Shortest-known faithful form stated? HCC legible?
- Verifiability: Hashes present? Expansion reproducible? Checkpoints and chain valid?
- Traceability: Summary/figure bindings present and resolvable? Bindings.csv consistent?
- Safety & Governance: Prereg, I/O policy, stability, ethics, and privacy all present?

Decision rule (minimum bar): Accept “AI-required” labeling only if all four criteria pass. Otherwise, request revision: either supply missing capsules/receipts/bindings or drop the “AI-required” claim and present a centaur-native (AI-optional) paper.

6. Blueprint Exemplars

Below are three end-to-end patterns that *genuinely* require AI expansion to comprehend: a DRAT-style SAT/SMT certificate (proof-length barrier), an 8D topology tour (representation barrier), and a 10^8-step rewrite proof (compositional + proof-length). Each includes a human-checkable claim (HCC), a capsule manifest (MRC), the AEVP cards to run, artifacts, and acceptance-criteria evidence.

6.1 MRP-DRAT-FACTOR (SAT/SMT certificate)

Barrier. Proof-length (gigantic unsat certificate).
HCC (plain). Under constraints R, no assignment exists satisfying CNF Φ (|clauses|≈10^6); thus property Q holds for all n ≤ N.

Capsule header (proof)


```
{
  "claim_id": "C1-UNSAT",
  "capsule_id": "MRC-DRAT-1",
  "version": "1.0.0",
  "kind": "proof",
  "hash": "sha256:…",
  "size_bytes": 2147483648,
```

"deps": [],

"resources": {"steps_max": 120000000, "ram_mb": 4096, "timeout_s": 3600, "threads_max": 4},

"nondeterminism": {"seed": 1729, "variance_bands": {"metric":"steps","sd_max":0}}

}

Inputs & constraints

{

"params": {"cnf_hash":"sha256:Φ…", "format":"DIMACS"},

"constraints": {"prereg_id":"osf.io/xyz12", "unsat_check":"DRAT"},

"io_policy": {"allow_browse": false, "allow_net": false, "allow_files": ["./Φ.cnf","./Φ.drat"]}

}

AEVP cards to run

- V-B Proof Replay (DRAT replay with checkpoint_freq=10^5 deletions/resolutions)
- V-E Units & Sanity (file hash + clause counts match header)
- V-F Receipts (model/version/seeds/sha_chain)
- V-I Citations & Binding (bind summary paragraphs to checkpoint IDs)

Artifacts (required)

- transcript.ndjson (resolution/deletion steps with goal/context hashes)
- checkpoints/chk_*.json (every 10^5 steps)
- summary.txt (10 bullets → 1-page explanation tied to checkpoints)
- receipts.json

Acceptance-criteria evidence

- Necessity: size_bytes and steps_max preclude human replay; Φ/DRAT hashes included.
- Verifiability: Any runner recomputes the DRAT hash chain; mismatch → minimal counterexample step.
- Traceability: Each bullet in summary.txt cites chk_k.
- Safety: No external I/O; prereg ID guards leakage.

Risks & limits

- If Φ.cnf is mutated, replay fails early (good).
- If provider drift changes parsing, receipts show version mismatch; use pinned toolchains.

6.2 MRP-8D-TOPO (persistent homology tour)

Barrier. Representation (8D structure; projections required).
HCC (plain). Dataset D exhibits a robust 8D 1-cycle ($\beta_1>0$ over scale interval I) aligned with variable V; effect persists under specified noise ε.

Capsule header (projection)


```
{
  "claim_id": "C2-TOPO",
  "capsule_id": "MRC-TOPO-8D",
  "version": "1.0.0",
  "kind": "projection",
  "hash": "sha256:…",
  "size_bytes": 524288000,
  "deps": [],
  "resources": {"steps_max": 2000000, "ram_mb": 8192, "timeout_s": 1800, "threads_max": 8},
  "nondeterminism": {"seed": 42, "variance_bands": {"metric":"Betti","sd_max":0}}
}
```


Inputs & constraints


```
{
  "params": {"dataset_ref":"doi:10.5281/zenodo.12345", "eps_noise": 0.01, "complex":"vietoris-rips", "max_dim": 2},
  "constraints": {"prereg_id":"osf.io/topo77", "split":"heldout_v2"},
  "io_policy": {"allow_browse": false, "allow_net": false, "allow_files": ["./D.parquet"]}
}
```

AEVP cards to run

- V-D Diagrammer (generate slice specs and narrated tour; enforce glossary labels)
- V-E Units & Sanity (scale ranges, sampling density)
- V-F Receipts
- V-G Transfer (toy cloud with planted loop)
- V-I Citations & Binding

Artifacts (required)

- metrics.json (Betti numbers vs. scale; stability under ε)
- slices/*.json (axis pairs, ranges, labels)
- tour.md (ordered viewing script; each step cites slice_id)
- summary.txt, receipts.json

Acceptance-criteria evidence

- Necessity: 8D structure cannot be captured statically; slices/ and tour.md prove representational need.
- Verifiability: Re-run yields identical metrics.json; minor numerical tolerance may be specified.
- Traceability: Every caption line in summary.txt points to slice_id and metrics at specified scales.
- Safety: No browsing; dataset license declared; privacy attested.

Risks & limits

- Projection cherry-picking: mitigated by V-D requiring the entire tour with fixed seeds.
- Numerical tolerances must be declared; otherwise spurious instability.

6.3 MRP-REWRITE-10^8 (rewrite confluence/termination)

Barriers. Proof-length + compositional (global invariants across many rules).
HCC (plain). Rewriting system R is confluent and terminating under ordering $\succ$; normalization reaches unique normal form for all inputs up to bound B.

Capsule header (proof + search)

{

"claim_id": "C3-REWRITE",

"capsule_id": "MRC-RWS-10e8",

"version": "1.0.0",

"kind": "proof",

"hash": "sha256:…",

"size_bytes": 1572864000,

"deps": [{"capsule_id":"MRC-CRITICAL-PAIRS","hash":"sha256:…"}],

"resources": {"steps_max": 100000000, "ram_mb": 6144, "timeout_s": 5400, "threads_max": 4},

"nondeterminism": {"seed": 1337, "variance_bands": {"metric":"steps","sd_max":0}}

}

Inputs & constraints

{

"params": {"order":"lexpath", "bound_B": 100000, "critical_pair_limit": 250000},

"constraints": {"prereg_id":"osf.io/rw999", "rules_hash":"sha256:R…"},

"io_policy": {"allow_browse": false, "allow_net": false, "allow_files": ["./R.rules","./pairs.idx"]}

}

AEVP cards to run

- V-B Proof Replay (critical pair generation + joinability proofs; checkpoint_freq=10^4)
- V-C Counterexample Search (bounded divergence up to B; emit smallest counterexample if found)
- V-E Units & Sanity (rule arities, ordering well-foundedness checks)
- V-F Receipts, V-I Binding, V-J Ethics

Artifacts (required)

- transcript.ndjson (joinability steps, termination measure decreases)

- checkpoints/ (hash chain over batches of pairs)
- coverage.json (explored terms up to B; coverage %)
- witnesses.json (null or minimal diverging pair)
- summary.txt, receipts.json

Acceptance-criteria evidence

- Necessity: steps_max=10^8 and critical_pair_limit prove beyond-human replay.
- Verifiability: Deterministic replay with hash chain; any mismatch produces a minimal non-joinable pair.
- Traceability: Summary binds per lemma to checkpoint_id and, when relevant, pair_id.
- Safety: No network; preregistered rules and bounds; optional gating if rules could be sensitive.

Risks & limits

- Over-aggressive bounds (B too small) undercut scope; prereg must fix B.
- If termination fails, V-C surfaces a counterexample; paper must present it honestly.

Quick comparison table (editor aid)

Exemplar	Barrier(s)	Core artifacts	Primary card(s)	Receipts must show
MRP-DRAT-FACTOR	Proof-length	transcript.ndjson, checkpoints/	V-B, V-F	steps≈1e8, sha_chain, deterministic
MRP-8D-TOPO	Representation	slices/*.json, tour.md, metrics.json	V-D, V-F, V-G	Betti curves, seeds, slice IDs
MRP-REWRITE-10^8	Proof-length + compositional	transcript, coverage.json, witnesses.json	V-B, V-C, V-F	steps≈1e8, coverage %, chain

Takeaway. These blueprints make “AI-required” concrete: each ships a human-checkable claim, but *full comprehension* (and verification) demands running AEVP to expand massive proofs,

explore huge state spaces, or tour high-dimensional structure—under receipts, checkpoints, and bindings that anyone can validate.

7. Authoring Workflow

This section is a how-to for producing a Machine-Required Paper (MRP). It splits responsibilities cleanly between the human-readable manuscript and the machine-expanded capsules, and it bakes in preregistration and leakage guards from day one.

7.1 Human narrative responsibilities

Purpose: Make the *claims* legible without running anything, and make the *substance* expandable and auditable.

Your job as author (checklist):

1. State Human-Checkable Claims (HCCs).
    - Template: Claim C_i: Under premises P and scope S, guarantee G holds; limits L.
    - Keep symbols small; define them in a local glossary.
2. Declare the barrier(s).
    - Proof-length, representation, compositional; one paragraph each explaining why the shortest faithful form exceeds human working memory.
3. Name the capsule(s) per claim.
    - "C_i is supported by MRC-7a (proof), depends on MRC-6f (search graph)."
4. Write layered summaries tied to checkpoints.
    - For each capsule, prepare summary.txt text (10 bullets → 1 page) with inline tags ⟨capsule_id, checkpoint_id⟩ placeholders that the runner will populate.
5. Specify what to run (AEVP Cards).
    - E.g., "Run V-B with checkpoint_freq=10^4; run V-I to bind summary; run V-F to emit receipts."

6. Own the framing and ethics.
    - Explain what the result means, where it fails, and why machine expansion is necessary; include dual-use notes and access scope.

Anti-patterns to avoid:

- Hand-wavy “trust us” paragraphs; diagrams without specs; numbers without capsule/step provenance.

7.2 Capsule construction

Goal: Build MRCs as sealed, content-addressed payloads ready for AEVP expansion.

A. Plan the graph

- Draw the capsule DAG: e.g., MRC-6f (search) → MRC-7a (proof) → {MRC-8c (projection), MRC-9d (audit)}.
- Assign kinds: proof | search | projection | audit | simulation | other.

B. Fill the header (see §3.1)

- claim_id, capsule_id, version, kind
- hash (sha256 over the payload bytes), size_bytes
- deps with exact hashes
- resources (steps_max, ram_mb, timeout_s, threads_max)
- nondeterminism (seed, variance bands)

C. Bundle inputs & I/O policy (see §3.2)

- params (problem bounds, evaluation settings)
- constraints (e.g., ledger hash, prereg ID)
- io_policy (allow_browse, allow_net, whitelisted files)

D. Define expected artifacts (see §3.3)

- Primary (e.g., transcript.ndjson), integrity (checkpoints/), narrative (summary.txt), attestation (receipts.json)

- Frequencies (checkpoint every N steps) and layer sizes (10 bullets / 200 words / 1500 words)

E. Attach verifiers & failure policy (see §3.4)

- Required cards (V-B, V-F, V-I minimum for proofs)
- failure_policy for checksum mismatch, timeout, missing dep

F. Build → hash → lock

- Produce the payload; compute hash; freeze deps by hash; write lockfile.json.
- Validate locally with a dry-run AEVP to generate sample receipts (on a small slice if needed).

G. Manuscript hooks

- Insert the header JSON (copy-safe) into the paper.
- Reference capsule IDs wherever summaries or figures appear.

Minimal authoring template (capsule directory)

```
MRC-7a/
  capsule.bin          # payload (proof/search/projection data)
  header.json          # header schema (IDs, hashes, resources)
  inputs.json          # params, constraints, io_policy
  artifacts.json       # expected artifacts manifest
  verifiers.json       # cards + failure policy
  lockfile.json        # deps with exact hashes
  README.md            # “how to run” (AEVP invocation)
```

7.3 Preregistration & leakage guards

Why: Prevent post-hoc tuning, accidental browsing, and evaluation on held-out data. Make expansions auditable.

A. Preregister before expansion

- Register: premises P, scope S, metrics, bounds (e.g., steps_max, B, L), capsule DAG, and I/O policy.
- Record IDs: include prereg_id in headers and the manuscript.

B. Split & seal

- Data partitions: fix train/validation/held-out (or “analysis” vs “confirmation”) with content hashes.
- Ledger/spec seals: store constraint files by hash; cite hashes in constraints.ledger.

C. I/O sandbox

- allow_browse=false, allow_net=false by default.
- Whitelist only local paths hashed in io_log. Any outbound I/O → FAIL.

D. Stability harness

- Fix jitter protocol (K runs; types of prompt perturbations).
- Report mean ± sd and ranking consistency (e.g., Kendall’s τ) in the manuscript or Repro Pack.

E. Drift sentinels

- Schedule tiny sentinel expansions weekly (fixed seeds and small slices).
- Log deltas in a public or appendix table (model/version, sha_chain changes).

F. Leakage checks

- Card V-J must assert no held-out access;
- Optional red-team card: attempt to induce leakage (e.g., prompt hints of held-out answers); report results.

G. Access & dual-use controls

- If artifacts could enable misuse, gate capsules (reviewer-only) and redact operational details from public summaries.
- State the policy in Ethics (who can run what and how logs are kept).

Submission-ready guardrail checklist

- Prereg ID in paper and headers
- Data/constraint hashes cited
- I/O policy forbids browsing/network (or explicitly allows with reason)
- Stability table (K, mean, sd, Kendall's τ)
- Drift sentinel plan noted
- Ethics report (risks, mitigations, access scope)
- Bindings present (summary → checkpoint)
- Example receipts included (even on a small slice)

Takeaway: Treat the human narrative as the contract, and the capsules as sealed evidence. With preregistration and leakage guards in place, editors and readers can expand, verify, and trust results that are intrinsically beyond unaided human comprehension.

8. Evaluation & Governance

Evaluation asks: *Do expansions give the same answers when they should, and change only for stated reasons?* Governance asks: *Can we run them responsibly?* This section specifies stability metrics, drift sentinels, and access/dual-use controls that editors can adopt verbatim.

8.1 Stability (mean±sd, ranking)

Purpose. Quantify variability of machine expansions under controlled prompt/model perturbations.

A. Protocol

- Unit of analysis. Any scalar or vector result derived from expansion (e.g., "improvement at α=0.8", "Betti curve at scale t", "#critical pairs joined").
- Jitter set. Choose K ≥ 5 prompt variants from a fixed menu (synonym swaps, punctuation, section order; no semantic changes). Keep capsule bytes, seeds, and version fixed unless testing *model* stability.
- Runs. Execute each variant once; collect outputs $y_1, \ldots, y_K$.

B. Metrics

- Point estimate. $\bar{y} = \frac{1}{K}\sum y_k$
- Dispersion. $s = \sqrt{\frac{1}{K-1}\sum(y_k - \bar{y})^2}$
- Coefficient of variation. $\mathrm{CV} = s/(|\ \bar{y}\ | + \delta)$, $\delta = 10^{-8}$to avoid division by zero.
- Interval (optional). $\bar{y} \pm 1.96 \cdot s/\sqrt{K}$(report K; avoid over-interpretation for tiny K).
- Ranking consistency (for multiple conditions). Compute Kendall's τ between each run's ranking $\pi_k$and the majority ranking $\pi^{\backslash *}$; report the median τ.

C. Decision thresholds (policy knobs)

- Scalar stability: CV ≤ 5% (strict venues) or ≤ 10% (general).
- Ranking stability: median Kendall's τ ≥ 0.90.
- Narrative guard: If CV > threshold but τ ≥ 0.90, authors must (i) report the large relative uncertainty and (ii) avoid over-precise claims.

D. Reporting template (drop-in)

Stability (K=5, jitter: synonym+punctuation):

metric: improvement@alpha=0.8

mean=0.208, sd=0.0041, CV=1.97% -> PASS

ranking (T' < T''): median Kendall's tau=1.00 -> PASS

Failure semantics. If stability fails, claims tied to the unstable metric must be downgraded (e.g., "suggestive") or re-established with stronger controls (pin model/version, remove fragile prompt cues).

8.2 Drift sentinels

Purpose. Detect unintended output changes caused by model/toolchain updates.

A. Sentinel design

- Scope. Select a *minimal, representative* slice per capsule: e.g., first 1e4 proof steps, one canonical 8D slice, or 1% of search space.
- Frozen inputs. Fix capsule bytes, seeds, and AEVP version; forbid browsing/network I/O.

- Schedule. Run weekly or on toolchain updates; keep last N=12 receipts.

B. Sentinel artifacts

- sentinel_receipts.jsonl: one line per run with model/version, sha_chain tail, walltime, steps.
- delta_report.md: highlights any change (hash mismatch, walltime spikes, metric drifts).

C. Drift thresholds & actions

- Hash/chain mismatch: Critical — treat as model/tool change; pin or re-run full expansions and update receipts.
- Metric delta: If scalar drift > 1 sd of stability band → annotate manuscript with a *drift note*; if > 3 sd → require re-evaluation and possibly version bump.
- Walltime/memory spikes: If > 2× historical median, file an efficiency anomaly (may indicate runtime/config drift).

Editor policy. Submissions must include one month of sentinel history (or a plan for post-acceptance archival). Camera-ready updates require re-running sentinels.

8.3 Access control & dual-use scoping

Purpose. Prevent harmful operationalization and ensure compliant data use while preserving auditability.

A. Access tiers

- Tier A (Public summaries). Manuscript text, summary.txt, non-operational figures, bindings.csv.
- Tier B (Reviewer capsules). Full MRC payloads + headers + verifiers; shared via signed links; audit logs enabled.
- Tier C (Redacted ops). Sensitive steps (e.g., parameterized exploitation routines) removed or gated; released only to vetted auditors under DUA.

Declaration block (in paper):

access:

public: [summary.txt, bindings.csv, receipts_example.json]

reviewer_only: [MRC-DRAT-1.bin, MRC-RWS-10e8.bin, verifiers.json]

gated: [MRC-TOOLING-OPS.bin]

audit_logs: enabled

B. I/O sandbox & policy enforcement

- Defaults: allow_browse=false, allow_net=false; whitelist only hashed local files.
- Receipts must log: every path read/written + hash; any network call → FAIL.
- PII/data licensing: Capsules must pass privacy_ok and include license strings; otherwise reject or require sanitization.

C. Dual-use review (V-J)

- Risk assessment: Identify operational pathways (e.g., generating exploit configs, enabling illicit synthesis).
- Mitigations: Redact/gate steps; publish only aggregated metrics; throttle or watermark outputs.
- Accountability: Keep an access ledger (who accessed which capsule, when); include contact for responsible disclosure.

D. Example decision matrix

Scenario	Risk	Action
DRAT unsat proof for benign combinatorics	Low	Tier A+B; full public after acceptance
8D topology on sensitive medical data	Privacy	Tier B only; de-identify; aggregate metrics; IRB letter
Rewrite system enabling obfuscated malware	Dual-use	Tier C; redact ops; vetted auditor access; watermarking

---

One-page editorial checklist

- Stability: K, jitter menu, mean±sd, CV, ranking τ reported? Thresholds met?
- Drift: Sentinel plan and last 4 receipts included? Any deltas explained?
- Access: Tiers declared? I/O policy enforced in receipts? PII/licensing clear?
- Dual-use: V-J ethics report present? Mitigations proportional to risk?

Takeaway. Stability quantifies reliability; sentinels watch for silent changes; and access controls let us publish beyond-human results safely—without giving up auditability or reproducibility.

9. Reader & Reviewer Guide

This guide turns the paper's contracts (HCC/MRC/AEVP) into a short, repeatable workflow. It minimizes tooling and maximizes audit value per minute.

9.1 Minimal steps to inspect (15–45 minutes)

A. Understand the claim (no machine needed) — 5 min

1. Read the Human-Checkable Claim (HCC): premises P, guarantee G, scope S, limits L.
2. Note the barrier(s) claimed (proof-length / representation / compositional).
3. Identify supporting capsules (capsule IDs) for this claim.

B. Verify the capsule identity — 3 min

1. Copy the capsule's header JSON from the manuscript.
2. Recompute the hash of the capsule file (if provided) and match it to header.hash.
3. Confirm deps[].hash exist and match (or are bundled).

C. Run the smallest AEVP slice — 7–15 min

1. Execute the paper's "How to run (slice)" command (provided in README or appendix).
2. Require V-F receipts and at least one integrity card (V-B replay or V-D diagrammer).
3. Inspect receipts.json: check model, version, seed, steps, sha_chain, io_log (no network unless allowed).

D. Bind summaries to artifacts — 5–10 min

1. Open summary.txt; pick any paragraph with ⟨capsule_id, checkpoint_id⟩.
2. Open the referenced checkpoint and locate the corresponding entry in transcript.ndjson or slices/*.json.
3. Confirm one-to-few mapping (no hand-wavy claims without anchors).

E. Spot stability/drift — 5–10 min

1. Read the stability table (mean±sd, CV, ranking τ).
2. Glance at drift sentinel receipts; compare last run vs. prior (hash tail, walltime).

Pass/hold rule: If any of (hash mismatch, unbound summary, missing receipts, network I/O against policy) occurs → HOLD and file an issue.

9.2 Issue taxonomy (file with capsule/checkpoint precision)

Use short codes so authors can triage quickly; always include capsule_id and, if applicable, checkpoint_id.

- N-1 Necessity gap. Barrier not justified; a human-length faithful form appears possible.
- V-1 Verifiability gap. Receipts missing/invalid; hash chain not reproducible; deps unresolved.
- T-1 Traceability gap. Summary or figure lacks binding to checkpoint_id / slice_id; numbers unlocatable.
- S-1 Stability gap. CV > threshold and/or median Kendall's τ < threshold; jitter protocol unclear.
- D-1 Drift anomaly. Sentinel hash/metrics changed beyond bands without an explained version bump.
- L-1 Leakage risk. Evidence of held-out use or off-policy browsing/network I/O.
- A-1 Access/ethics risk. Dual-use not mitigated; missing IRB/license; privacy_ok not attested.
- U-1 Units/sanity error. Dimensional inconsistency; impossible ranges.
- C-1 Counterexample. V-C found a witness within declared bounds; scope must be corrected.
- R-1 Resource claim mismatch. Steps/memory/time exceed header without truncated status.

Report format (example):

issue: T-1

capsule_id: MRC-DRAT-1

checkpoint_id: chk_0042

evidence: summary.txt ¶3 cites chk_0042, but transcript.ndjson has no matching step hash.

severity: block

suggestion: re-bind summary or regenerate bindings.csv

9.3 Repro Pack contents (submission checklist)

A minimal, self-contained bundle enabling any reviewer to validate identity, run a slice, and check bindings.

A. Core

- README.md — one-page quick start with slice command (no network).
- headers/*.json — capsule headers copied from the manuscript.
- capsules/*.bin — capsule payloads (or small surrogates for a slice).
- lockfile.json — exact deps with hashes.
- io_policy.json — allow_browse/allow_net flags and allowed local paths.

B. Verifiers & configs

- verifiers.json — required AEVP cards + failure policy.
- config.json — checkpoint frequency, seeds, tolerance bands (if any).
- glossary.json — canonical labels for terms/axes (prevents diagram label drift).

C. Run outputs (expected)

- expected/receipts_example.json — from a successful slice (abridged).
- expected/bindings.csv — claim→capsule→checkpoint→ref mapping.
- expected/summary.txt — layered narrative with checkpoint tags.
- (By exemplar)

    - DRAT: expected/transcript.ndjson, expected/checkpoints/chk_*.json
    - 8D: expected/metrics.json, expected/slices/*.json, expected/tour.md
    - Rewrite: expected/coverage.json, expected/witnesses.json (null or smallest witness)

D. Governance

- prereg.txt — registry ID(s), date, scope, metrics.
- stability.md — K, jitter menu, mean±sd, CV, ranking τ.
- sentinel.jsonl — last N runs (model/version, sha tail, walltime).
- ethics_report.md — dual-use risks, redactions, access tiers.
- licenses/ — data/code licenses; privacy_ok attestation.

E. Sanity scripts (optional but ideal)

- scripts/check_hashes.py — recompute capsule/deps; verify lockfile.
- scripts/validate_bindings.py — confirm every summary tag resolves to a checkpoint/slice.
- scripts/run_slice.sh — single-command AEVP invocation for the smallest demonstration.

Acceptance tip: If a submission lacks any item above, request targeted revision: add the missing receipts/bindings or withdraw the “AI-required” label and resubmit as a centaur-native (AI-optional) paper.

10. Risks, Ethics, and Limitations

This section names the principal hazards of Machine-Required Papers (MRPs), pairs each with mitigations and audit hooks, and clarifies what MRPs cannot guarantee. Editors can lift these items directly into policy.

10.1 Risks

R1. False authority via inscrutable expansion

Risk. Long transcripts and hash chains can *look* convincing while hiding subtle specification bugs or bad modeling choices.
Mitigations.

- Require Necessity + Verifiability + Traceability (Section 5) for the “AI-required” label.
- Enforce bindings: every summary paragraph cites {capsule_id, checkpoint_id}.
- Mandate V-I Citations and V-B Proof Replay; fail fast on mismatch.

R2. Model drift misleading readers

Risk. Provider updates alter outputs; the same capsule yields different narratives.
Mitigations.

- Receipts must log model/version/date; pin frozen endpoints for archival runs.
- Run drift sentinels (Section 8.2) and publish deltas; version-bump on material change.

R3. Data leakage and post-hoc tuning

Risk. Constraints or prompts accidentally encode held-out knowledge.
Mitigations.

- Preregistration of constraints/splits/metrics; reference IDs in headers.
- I/O sandbox: allow_browse=false, allow_net=false; log all paths + hashes in receipts.
- Optional red-team card to detect leakage patterns.

R4. Dual-use operationalization

Risk. Expanded artifacts enable misuse (e.g., exploit generation, sensitive synthesis).
Mitigations.

- Access tiers (public/reviewer/gated) with audit logs (Section 8.3).
- Redact or gate operational steps; publish aggregates instead of runnable details.
- Require V-J Ethics analysis proportional to risk.

R5. Privacy and licensing violations

Risk. Capsules embed PII or unlicensed data; projections reveal sensitive structure.
Mitigations.

- Capsule hygiene: no PII; licensed datasets only; include license text.
- PII scanners pre-expansion; receipts set privacy_ok: true.
- For sensitive domains, IRB or equivalent oversight.

R6. Over-fitting the format, under-reporting the science

Risk. Authors satisfy protocol checkboxes while making weak or trivial claims.
Mitigations.

- Editors evaluate substantive novelty separately from protocol compliance.
- Require a human-legible HCC with clear scientific stakes.

R7. Reproducibility vs. resource inequality

Risk. Expansions need compute some reviewers lack.
Mitigations.

- Provide slice runs (small checkpoints or partial tours) that validate identity and bindings.
- Encourage community re-execution pools or venue-hosted runners for heavy capsules.

R8. Security of capsule execution

Risk. Malicious payloads or unsafe code paths.
Mitigations.

- Treat capsules as data, not code; runners must not execute embedded binaries.
- Air-gapped verification option; strict MIME/format validation; content hashing.

10.2 Ethical Principles

1. Necessity over spectacle. Use “AI-required” only when barriers (proof-length, representation, compositional) genuinely apply.
2. Least privilege comprehension. Reveal only what is needed for understanding and verification; gate the rest.
3. Layered transparency. Summaries first; deeper layers on demand; every layer bound to checkpoints.
4. Accountable automation. The machine does work; the authors remain responsible for claims, scope, and ethics.

5. Proportionality. Dual-use mitigations scale with risk; benign combinatorics ≠ bio-ops.

10.3 Limitations (of MRPs and this framework)

- L1. Protocol compliance ≠ truth. A perfectly verified expansion can still formalize the *wrong* model or irrelevant assumptions. MRPs certify *integrity of execution*, not correctness of scientific worldview.
- L2. Dependence on toolchains. Receipts reduce but don't remove reliance on specific solvers/models; long-term archival may require emulation.
- L3. Numeric fragility. High-D projections and large proofs can be sensitive to numerical tolerances; bands must be declared and may still be disputed.
- L4. Compute & time costs. Full expansions may be impractical for all reviewers; slice runs mitigate but don't replace deep audits.
- L5. Accessibility. Even with layered summaries, some readers may find AI-narrated artifacts challenging; journals should offer non-interactive pedagogical summaries.
- L6. Evolving standards. Hash algorithms, runner frameworks, and model endpoints will change; receipts and schemas must version appropriately.

10.4 Ethics/Compliance Checklist (submission-ready)

- Human-Checkable Claim (P/G/S/L) stated in prose/math
- Barrier justification (proof-length / representation / compositional)
- Capsule headers with hashes, resources, deps
- Receipts example (even for a slice)
- Bindings (summary → checkpoint / slice)
- Preregistration ID(s); data/constraint hashes
- I/O sandbox policy; io_log shows only allowed paths
- Stability report (K, mean±sd, CV, ranking τ)
- Drift sentinel history or plan
- Ethics report: dual-use mitigations, access tiers, licenses, privacy_ok

- Security note: capsules are data; runner does not execute embedded code

---

10.5 Response Plan (when things go wrong)

- Hash mismatch / replay failure: Halt publication pipeline; request capsule rebuild; attach "delta note" with new hash/version.
- Leakage evidence: Suspend "AI-required" label; redo prereg & partitions; re-expand in sandbox.
- Drift detected (beyond bands): Pin tools; re-run expansions; update receipts; annotate manuscript.
- Dual-use concern raised: Move sensitive capsules to gated Tier C; convene rapid ethics review; publish sanitized summaries.

Takeaway. MRPs push comprehension beyond human working memory while keeping results auditable, responsible, and limited where necessary. Clear risks exist—but with disciplined protocol, preregistration, receipts, and governance, those risks become *manageable and reviewable*, not hidden.

11. Implementation Toolkit

This toolkit makes the paper runnable: copy-safe JSON schemas, minimal checker pseudo-code, and reference receipts you can adapt. Schemas are intentionally small; extend by versioning (additive fields only), never by changing meanings.

11.1 JSON schemas (copy-safe)

A) Capsule header (header.json)

```
{
  "schema": "mrc.header@1.0.0",
  "claim_id": "C1",
  "capsule_id": "MRC-7a",
  "version": "1.0.0",
  "kind": "proof|search|projection|audit|simulation|other",
```

```
  "hash": "sha256:...payload-bytes...",
  "size_bytes": 0,
  "deps": [
    {"capsule_id":"MRC-6f","hash":"sha256:...","version":"1.0.0"}
  ],
  "resources": {"steps_max":0,"ram_mb":0,"timeout_s":0,"threads_max":1},
  "nondeterminism": {"seed":0,"variance_bands":{"metric":"steps","sd_max":0}},
  "provenance": {"authors":[],"created":"YYYY-MM-DD","license":"CC BY 4.0"}
}
```


B) Inputs & policy (inputs.json)


```
{
  "schema": "mrc.inputs@1.0.0",
  "params": {"alpha":0.8,"bounds":{"N":100000,"L":100000}},
  "constraints": {"ledger_hash":"sha256:...","prereg_id":"osf.io/abcd1"},
  "io_policy": {"allow_browse": false, "allow_net": false, "allow_files": ["./capsules/MRC-6f.bin"]}
}
```


C) Expected artifacts (artifacts.json)


```
{
  "schema": "mrc.artifacts@1.0.0",
  "artifacts": [
    {"name":"transcript.ndjson","role":"primary","required":true},
    {"name":"checkpoints/","role":"integrity","required":true,"freq":10000},
    {"name":"summary.txt","role":"narrative","required":true,"layers":[10,200,1500]},
    {"name":"receipts.json","role":"attestation","required":true}
  ]
}
```

D) Verifiers & failure policy (verifiers.json)

```
{
  "schema": "mrc.verifiers@1.0.0",
  "verifiers": [
    {"card":"V-B","params":{"checkpoint_freq":10000}},
    {"card":"V-F","params":{"receipts_required":true}},
    {"card":"V-I","params":{"bind_summary_to_checkpoints":true}}
  ],
  "failure_policy": {
    "on_checksum_mismatch":"fail-fast",
    "on_timeout":"emit_truncated_receipt",
    "on_missing_dep":"abort_with_reason"
  }
}
```

E) Bindings (narrative ↔ artifacts) (bindings.csv)

```
claim_id,capsule_id,checkpoint_id,ref_id,page_line,notes
C1,MRC-7a,chk_0042,Ref17,"p12:L18-26","lemma A applied"
```

F) Glossary (glossary.json)

```
{
  "schema": "mrc.glossary@1.0.0",
  "terms": [
    {"term":"HCC","definition":"Human-Checkable Claim","first_seen":"§2.1"},
    {"term":"MRC","definition":"Machine-Required Capsule","first_seen":"§3"}
  ]
}
```

G) Receipts (receipts.json) — schema in §11.3 (below)

11.2 Minimal checker pseudo-code

Goal: a 150–250 line utility that any reviewer can run locally. Below is language-agnostic pseudo-code; port to Python/Go/Rust.

A) Hash & dependency check

```
function check_capsule_identity(header_path, capsule_path, deps_dir):
  H := read_json(header_path)
  payload_hash := sha256_file(capsule_path)
  assert "sha256:"+payload_hash == H.hash

  for dep in H.deps:
    dep_file := path_join(deps_dir, dep.capsule_id + ".bin")
    assert file_exists(dep_file)
    assert "sha256:"+sha256_file(dep_file) == dep.hash
  return OK
```

B) Checkpoint chain verification

```
function verify_chain(checkpoints_dir, capsule_hash, seed, version):
  h0 := sha256("AEVP|root|" + capsule_hash + "|" + seed + "|" + version)
  expect := h0
  prev_chk := h0
  list := sort_naturally(list_files(checkpoints_dir, "chk_*.json"))

  for chk_file in list:
    chk := read_json(chk_file)
    steps := read_step_hashes_between(chk.prev_idx, chk.step_idx)  # from transcript.ndjson
    h_steps := sha256_concat("AEVP|steps|", steps)
    h_chk := sha256(prev_chk + "|" + h_steps)
```

```
    assert h_chk == chk.sha_chain_tail
    prev_chk := h_chk
  return prev_chk   # tail of chain
```

C) Summary binding validation

```
function validate_bindings(summary_txt, transcript_idx, bindings_csv):
  B := read_csv(bindings_csv)
  T := build_index(transcript_idx)  # map checkpoint_id -> step range / hashes
  missing := []
  for row in B:
    if not T.contains(row.checkpoint_id):
      missing.append(row)
  assert len(missing) == 0
  # Optional: grep summary for all '⟨capsule_id, checkpoint_id⟩' tags
  tags := extract_tags(summary_txt)
  for tag in tags:
    assert T.contains(tag.checkpoint_id)
  return OK
```

D) I/O policy enforcement (receipts)

```
function enforce_io_policy(receipts_json, allowed_paths, allow_net):
  R := read_json(receipts_json)
  for io in R.io_log:
    assert normalize(io.path) in allowed_paths
  if allow_net == false:
    assert not any(is_network_event(e) for e in R.io_log)
  return OK
```

E) Stability quick check

```
function stability_report(values):  # values = [y1..yK]
  mean := avg(values)
  sd := std(values)
  cv := sd / max(abs(mean), 1e-8)
  return {"mean":mean,"sd":sd,"cv":cv}
```

CLI sketch

```
mrc-check \
  --header headers/MRC-7a.header.json \
  --capsule capsules/MRC-7a.bin \
  --deps ./capsules/ \
  --artifacts ./artifacts \
  --bindings expected/bindings.csv \
  --summary expected/summary.txt \
  --receipts expected/receipts.json \
  --allow-net false \
  --allow-path ./capsules/MRC-6f.bin
```

Exit codes

- 0 OK; 10 identity fail; 20 chain fail; 30 bindings fail; 40 I/O policy fail.

11.3 Reference receipts

Three abridged examples matching the blueprint exemplars. Replace … with real hashes.

A) DRAT unsat proof (MRP-DRAT-FACTOR)

```
{
  "capsule": {
    "capsule_id":"MRC-DRAT-1",
    "hash":"sha256:ab17…e2",
```

```
  "kind":"proof",
  "version":"1.0.0",
  "deps":[]
},
"runner": {
  "model":"GPT-5 Thinking",
  "version":"2025-10-06",
  "framework":"aevp/1.2.3",
  "hardware":{"cpu":"x86_64","ram_mb":32768,"gpu":"none"}
},
"execution": {
  "start_utc":"2025-10-06T14:12:03Z",
  "end_utc":"2025-10-06T15:10:07Z",
  "walltime_s":3494.0,
  "steps":118734211,
  "mem_mb_peak":2812,
  "threads":4,
  "checkpoint_freq":100000
},
"integrity": {
  "seed":1729,
  "sha_chain":["h0","h1","…","h1187"],
  "determinism":"deterministic",
  "variance_bands":null
},
"io_log": [
```

{"path":"./Φ.cnf","sha":"sha256:c0f…","mode":"read"},

{"path":"./Φ.drat","sha":"sha256:9aa…","mode":"read"},

{"path":"./artifacts/checkpoints/chk_1187.json","sha":"sha256:8de…","mode":"write"}

],

"artifacts": {

"primary":["transcript.ndjson"],

"integrity":["checkpoints/"],

"narrative":["summary.txt"],

"reports":["replay_report.txt"]

},

"status":{"state":"verified","reason":null,"first_failure_checkpoint":null},

"governance":{"prereg_id":"osf.io/xyz12","privacy_ok":true,"allow_browse":false,"allow_net":false}

}

B) 8D topology tour (MRP-8D-TOPO)

{

"capsule":{"capsule_id":"MRC-TOPO-8D","hash":"sha256:f3a…","kind":"projection","version":"1.0.0","deps":[]},

"runner":{"model":"GPT-5 Thinking","version":"2025-10-06","framework":"aevp/1.2.3","hardware":{"cpu":"x86_64","ram_mb":65536,"gpu":"none"}},

"execution":{"start_utc":"2025-10-06T16:00:00Z","end_utc":"2025-10-06T16:28:45Z","walltime_s":1725,"steps":2048112,"mem_mb_peak":7430,"threads":8,"checkpoint_freq":10000},

"integrity":{"seed":42,"sha_chain":["h0","…","h205"],"determinism":"numeric_tolerances","variance_bands":{"metric":"Betti","sd_max":0}},

"io_log":[{"path":"./D.parquet","sha":"sha256:77b…","mode":"read"}],

"artifacts":{"primary":["metrics.json","tour.md"],"integrity":["checkpoints/"],"narrative":["summary.txt"],"reports":[]},

"status":{"state":"verified","reason":null,"first_failure_checkpoint":null},

"governance":{"prereg_id":"osf.io/topo77","privacy_ok":true,"allow_browse":false,"allow_net":false}

}

C) Rewrite confluence/termination (MRP-REWRITE-10^8)

{

"capsule":{"capsule_id":"MRC-RWS-10e8","hash":"sha256:5bc…","kind":"proof","version":"1.0.0","deps":[{"capsule_id":"MRC-CRITICAL-PAIRS","hash":"sha256:1dd…"}]},

"runner":{"model":"GPT-5 Thinking","version":"2025-10-06","framework":"aevp/1.2.3","hardware":{"cpu":"x86_64","ram_mb":24576,"gpu":"none"}},

"execution":{"start_utc":"2025-10-06T18:02:10Z","end_utc":"2025-10-06T19:32:40Z","walltime_s":541,
"steps":100000000,"mem_mb_peak":6120,"threads":4,"checkpoint_freq":10000},

"integrity":{"seed":1337,"sha_chain":["h0","…","h10000"],"determinism":"deterministic","variance_bands":null},

"io_log":[

{"path":"./R.rules","sha":"sha256:aa1…","mode":"read"},

{"path":"./pairs.idx","sha":"sha256:bb2…","mode":"read"}

],

"artifacts":{"primary":["transcript.ndjson","coverage.json","witnesses.json"],"integrity":["checkpoints/"],"narrative":["summary.txt"],"reports":["units_report.md"]},

"status":{"state":"verified","reason":null,"first_failure_checkpoint":null},

"governance":{"prereg_id":"osf.io/rw999","privacy_ok":true,"allow_browse":false,"allow_net":false}

}

Usage notes

- Version everything. Schema strings (mrc.header@1.0.0) let tools reject incompatible bundles.
- Treat capsules as data. Runners parse and replay; they do not execute embedded code.
- Fail fast, explain precisely. On any mismatch, emit the offending checkpoint and a human-legible reason.

Takeaway: With these schemas, a 200-line checker, and example receipts, editors and reviewers can validate identity, integrity, and bindings in minutes—and authors can ship beyond-human results that are still auditable, reproducible, and safe.

12. Conclusion & Future Standards

Machine-Required Papers (MRPs) acknowledge a simple fact: some valid arguments are only accessible through machine expansion. Our framework preserves human responsibility—clear claims, ethics, limits—while enforcing verifiable AI execution through capsules, checkpoints, and receipts. The point is not to romanticize opacity; it is to discipline it so that beyond-human content is auditable, reproducible, and safe.

12.1 What this gives the community

- A contract, not a vibe: Human-Checkable Claims (HCCs) + Machine-Required Capsules (MRCs) + AEVP yields a testable pact between authors and readers.
- Interchangeability: Content-addressed hashes, schema-versioned JSON, and card-based verification let different runners produce comparable receipts.
- Governance hooks: Stability metrics, drift sentinels, I/O sandboxing, and access tiers make policy enforceable rather than aspirational.

12.2 Minimum Viable Standard (MVS) for venues (adopt today)

- Labeling: Use “AI-required” only if Sections 5.1–5.4 all pass (Necessity, Verifiability, Traceability, Safety).
- Artifacts: At least one capsule per flagship claim with header.json, inputs.json, artifacts.json, verifiers.json, lockfile.json.
- Receipts: Provide a slice run (small but real) with receipts.json, checkpoints/, and summary.txt bound to checkpoint_ids.
- Stability: Report mean±sd and ranking τ for K≥5 jittered runs.
- Drift: Include a 4-week sentinel history or a post-acceptance plan.
- Ethics: I/O policy, preregistration ID(s), access tiers, and privacy/licensing statements.

Badge (MVS-2025): Issued when all above pass; visible on the PDF and landing page.

12.3 Ambitious v2 Standard (12–24 months)

- Inter-runner reproducibility: At least two independent AEVP runners must recreate the chain tail and receipts.
- Persistent registry: DOIs for capsules and signed hash manifests; immutable public logs of receipts and sentinel deltas.
- Differential audit: Tools that compute and publish semantic diffs between receipts (model drift, tolerance changes).
- Curated benchmarks: Domain bundles (DRAT, 8D-TOPO, REWRITE-10^8) with gold receipts for continuous integration at journals.
- Access escrow: Standard reviewer gating and accountability (access ledgers, redaction templates).

Badge (MRP-Gold): Adds inter-runner proof and registry compliance to MVS.

12.4 Sample policy text (drop-in for CFPs)

AI-Required Submissions. Authors claiming “AI-required” must (i) justify a barrier (proof-length, representation, or compositional); (ii) supply Machine-Required Capsules with headers, dependency hashes, and verifiers; (iii) provide slice receipts (model/version/seed, step counts, hash chain); (iv) bind all MES-derived text to ⟨capsule_id, checkpoint_id⟩; (v) report stability (mean±sd; ranking τ); (vi) include preregistration IDs, I/O policy, ethics notes, and access tiers.

Submissions failing any item will be treated as AI-optional (centaur-native) or returned for revision.

12.5 Archival & interoperability

- Persistent identifiers: Assign a DOI to each capsule *and* its header; store payloads in content-addressed repositories.
- Schema governance: Version strings like mrc.header@1.0.0; only additive changes between minor versions.
- Long-term readability: Runners must treat capsules as data; avoid executable blobs; prefer open formats (NDJSON, CSV, Markdown, JSON).

12.6 Adoption roadmap

Next 3 months (pilot):

- 2–3 venues adopt MVS-2025; authors ship one capsule with slice receipts; reviewers run the minimal checker.

6–12 months (scale):

- Community registry launches (hash/DOI index + sentinel logs).
- First cross-runner reproduction workshop; publish inter-runner deltas.

12–24 months (standardize):

- v2 badge criteria (MRP-Gold) enforced at top venues.
- Benchmarks and CI bots validate receipts on submission.

12.7 Key success metrics

- Compliance rate: % of AI-required papers passing MVS on first review.
- Repro rate: % of capsules whose slice receipts are reproduced by third parties.
- Drift detection: median time from provider change → sentinel alert.
- Safety incidents: number of dual-use/privacy violations (target: zero).
- Reader utility: survey scores on clarity of HCCs and usefulness of bound summaries.

12.8 Open questions

- Numerical tolerances: How to standardize bands for floating-point heavy expansions without incentivizing slack?

- Privacy-preserving verification: Can zero-knowledge or secure enclaves certify expansions without exposing raw data?
- Attribution & credit: How do we cite capsule builders, runner maintainers, and auditors fairly?
- Costs & equity: Who funds compute for deep audits; can registries subsidize replication?

12.9 Final takeaway

MRPs do not replace human judgment; they restore it by separating what must be read by people from what must be expanded and checked by machines—under receipts and constraints anyone can verify. With a minimal standard today and a clear path to robust cross-runner reproducibility, the community can publish results that are *beyond human working memory* without sacrificing rigor, safety, or trust.

References

[1] Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*.

[2] Lipton, Z. C. (2016/2018). The mythos of model interpretability. *arXiv:1606.03490*; updated in *Communications of the ACM*, 61(10), 36–43 (2018).

[3] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. *KDD*.

[4] Jacovi, A., & Goldberg, Y. (2020). Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? *ACL*.

[5] Heule, M. J. H., Wetzler, N., & Hunt Jr., W. A. (2015). DRAT-trim: Efficient checking and trimming using extended resolution. *SAT*.

[6] Cook, S. A., & Reckhow, R. A. (1979). The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1), 36–50.

[7] Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.). (2009). Handbook of Satisfiability. IOS Press.

[8] de Moura, L., & Bjørner, N. (2008). Z3: An efficient SMT solver. *TACAS*.

[9] de Moura, L., Kong, S., Avigad, J., van Doorn, F., & von Raumer, J. (2015). The Lean theorem prover (system description). *CADE*.

[10] The Coq Development Team. (2008+). The Coq Proof Assistant Reference Manual. INRIA.

[11] Wetzler, N., Heule, M. J. H., & Hunt Jr., W. A. (2014/2016). The DRAT format and efficient SAT proof checking. *SAT Competition Workshop/Journal versions*.

[12] Edelsbrunner, H., & Harer, J. (2010). Computational Topology: An Introduction. AMS.

[13] Cohen-Steiner, D., Edelsbrunner, H., & Harer, J. (2007). Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1), 103–120.

[14] Ghrist, R. (2008). Barcodes: The persistent topology of data. *Bulletin of the AMS*, 45(1), 61–75.

[15] Bauer, U. (2021). Ripser: Efficient computation of Vietoris–Rips persistence barcodes. *Journal of Applied and Computational Topology*, 5, 391–423.

[16] Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2), 81–93.

[17] Lamport, L. (1981). Password authentication with insecure communication. *Communications of the ACM*, 24(11), 770–772. (Classical hash chain reference.)

[18] Merkle, R. C. (1987). A digital signature based on a conventional encryption function. *CRYPTO* (introduces Merkle trees for content-addressed integrity).

[19] Pineau, J., et al. (2020). Improving reproducibility in machine learning research: A report from the NeurIPS 2019 reproducibility program. *Journal of Machine Learning Research*, 22(164), 1–20. (Includes the Reproducibility Checklist.)

[20] Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11), 2600–2606.

[21] Benet, J. (2014). IPFS — Content addressed, versioned, P2P file system. *arXiv:1407.3561*.

[22] Park, J. S., Hendrycks, D., Mazeika, M., & Song, D. (2023). In defense of pre-training for out-of-distribution detection. *ICLR*. (Representative for drift/sentinel monitoring discussion in ML.)

[23] Holtz, M., et al. (2022). Best practices for scientific computing with data provenance and audit trails. *Patterns*, 3(7), 100555. (Provenance & I/O logging concepts.)

[24] O'Leary, D. P. (1990). Scientific computing with case studies. *SIAM Review*, 32(4), 690–694. (Classic note on stability and numerical sensitivity.)

[25] Ribeiro, M. T., et al. (2020). Beyond accuracy: Behavioral testing of NLP models with CheckList. *ACL*. (Template for issue taxonomy & audit-style evaluation.)

Notes.
• Items [11], [22]–[25] are representative anchors for practices (proof checking, drift/stability, audit-style evaluation) rather than the only possible citations; swap for domain-specific sources as needed.