

Adaptive Counterexample-Guided Refinement for Mathematics (ACGR-M)

A General, Human-Lightweight Protocol for Iterative Mathematical Discovery

Version 3.0 – Release Candidate 1 – August 2026

Full-Context Research, Assurance, Promotion, and Role-Separated Adversarial Review

Guillaume Bolduc

Abstract

ACGR-M is a research protocol for attacking difficult mathematical problems by separating discovery, exact acceptance, regression protection, structured failure harvesting, representation change, abstraction, and proof. A research harness searches for constructions, proofs, certificates, counterexamples, and informative residuals; accepted mathematics is validated exactly; failures and dead ends are preserved as first-class evidence; and each new generation is justified by the mechanisms exposed by the previous one.

Version 3.0 preserves the complete Version 2.7 scientific engine and adds enforceable governance derived from live campaigns: three non-interchangeable roles, immutable review objects, separate ruling/finding/presentation states, blocking predicates, computation thresholds, split scientific/handoff/transfer closure, external-review governance, and deterministic release identity. These additions strengthen rather than weaken every Version 2.7 correctness boundary.

Version 2.7 deliberately restores the full decision context of Version 2.3 and layers the semantic and assurance safeguards of Version 2.6 on top of it. Testing with an ACGR-M-naive agent showed that the lean Version 2.6 document was correct as a protocol summary but insufficient as a sole onboarding artifact: it omitted enough of the detailed reasoning, examples, operational invariants, research heuristics, and failure-class distinctions from Version 2.3 that initial research decisions degraded. Version 2.7 therefore treats the detailed Version 2.3 guidance as normative decision support rather than as historical verbosity. It preserves the full mature architecture of Versions 2.1–2.3 while adding Version 2.6 claim-evidence binding, raw-evidence validation, consequence-sensitive and applicability-closed assurance, typed outcomes, adversarial verifier mutation testing, operator-surface closure, terminal sealing, bounded cancellation, result promotion, evidence-minimal independent audit, role-separated environments, and semantic migration discipline.

Version 2.7 also makes **agent-continuity transfer** a first-class operational requirement for AI-mediated campaigns. A fresh agent must not be expected to reconstruct campaign history, negative knowledge, operator constraints, and scientific rationale from a short status summary. Each generation should therefore provide a one-click, read-only way to build a **transfer ZIP** containing the full governing protocol, a detailed transfer brief, relevant immutable project and result artifacts, and a manifest, while deliberately excluding the scientific handoff archive itself. The handoff remains the authoritative completed-run evidence; the transfer ZIP preserves the intellectual and operational context needed by a new agent instance to interpret that evidence correctly and continue without restarting the campaign.

The intended human interface remains deliberately small: initialize once, run the currently authorized generation, optionally inspect read-only status, stop only on a clear signal or external need, return the handoff package, install the next isolated generation, and resume. The agent is responsible for the mathematics, research strategy, diagnostics, code generation, verification design, refinement, and routine generation transitions. At every generation it must seek stronger mathematical progress and better efficiency, but **never by weakening correctness, exact acceptance, claim-evidence binding, regression protection, certificate integrity, provenance, reproducibility, human auditability, or proof quality**.

Contents

1	Core Principle	8
2	Version Lineage: What Version 2.3 Established and Version 2.7 Preserves	9
3	Normative Language	12
4	What the Agent Is Asked to Do	12
5	Cold-Start Orientation for a Fresh or Context-Poor Agent	13
6	Suitability Check Before Search	14
7	Three-Layer Research Model	14
8	Mathematical and Campaign State	15
9	Claim Characterization and Canonical Claim/Assurance Record	16
10	Claim Registry and Evidence Dependency Closure	17
11	Claim-Bound Exact Acceptance	18
12	Trusted Computing Base and Verifier Independence	19
13	Consequence-Sensitive Assurance and Gate Closure	20
14	Counterexamples, Residuals, and Negative Knowledge	21
15	The Mature ACGR-M Loop	23
16	Scientific Generation Contract	24
17	Representation Pressure	25
18	Discovery, Compression, and Proof	26
19	Compression Soundness and Coverage	26
20	Targeted Challenge Generation and Novelty	27
21	Adaptive Learning and Counterfactual Utility	27
22	Exact Acceptance and Verifier Diversity	28
23	Verifier/Oracle Incidents, Quarantine, and Repair	29
24	Numerical and Objective Discipline	30
25	Regression Protection and Semantic Equivalence	30

26 Reproducibility Classes and Controlled Nondeterminism	31
27 Finite Exhaustion, Minimality, and Negative Claims	31
28 Generations, Mechanisms, and Campaigns	32
29 Stop Signals and Handoff Criteria	33
30 Benchmarking and Information Efficiency	34
31 Exploratory Discovery and Confirmatory Evaluation	35
32 Benchmark Governance and Contamination	35
33 Ablations and Causal Attribution	36
34 Generation Ledger and Immutable Lineage	36
35 One Implementation per Generation	38
36 Human-Lightweight Project Layout	38
37 Project Identity, Hash Pinning, and State Manifest	40
38 Schema Versioning and Migration Discipline	40
39 Safe Launch and Runtime Invariants	41
40 Runtime Accounting	42
41 Transactional Checkpoints, Worker Isolation, and Commit Discipline	42
42 Forced Interruption and Recovery Testing	43
43 Release-Artifact and Operator-Surface Closure	44
44 Automatic Handoff Package	46
45 Agent-Continuity Transfer ZIP	47
46 Mathematical-Result Promotion and Skeptical Human Audit	50
47 Evidence Retention, Compaction, and Cold Storage	51
48 What the Refining Agent Must Do With the Handoff	51
49 Mandatory Efficiency Review	53
50 Resource Allocation Across Research Lanes	54
51 Branches, Competing Representations, and Merge Discipline	54
52 Generation Release Gate – A1 Detailed Acceptance	55

53 Claim Promotion Gate – A2	57
54 Independent Audit Gate – A2/A3	57
55 Publication and Novelty Gate – A3	58
56 Independent Reconstruction and Clean-Room Replication	58
57 Proof Red-Team Phase	59
58 External Frontier Provenance	60
59 Protocol Evolution Across Campaigns	61
60 Human’s Entire Job	61
61 Agent Autonomy Contract	62
62 Optional Proof-Complete Branch	63
63 Minimal Machine-Readable Refinement Signal	63
64 Recommended Machine-Readable Generation Entry	64
65 Final Instruction Template	65
66 Version 3.0 Governance and Quality System	67
66.1 Release identity and route to the author	67
66.2 Release-candidate boundary	67
67 Standing Instructions for ACGR-M v3.0	67
67.1 Preservation	67
67.2 Authority and roles	68
67.3 Review	68
67.4 Computation	68
67.5 Evidence and status	68
67.6 Packaging	69
68 Canonical Source Graph for ACGR-M v3.0	69
68.1 Canonical nodes	69
68.2 Generated nodes	70
68.3 Source registry record	70
68.4 Clause completeness and equivalence	70
68.5 Deterministic build requirements	71
68.6 Authority failure behavior	71
69 State-Transition Invariants for ACGR-M v3.0	71
69.1 Axes	71
69.2 Event requirements	72
69.3 Core invariants	72
69.4 Transition authority	73

69.5	Forbidden combinations	73
69.6	Decision lattice for hostile review	73
69.7	Required fixtures	74
70	ACGR-M v3.0 Supervisor Role	74
70.1	Owns	74
70.2	Required startup	75
70.3	Required conduct	75
70.4	Prohibited	75
70.5	Outputs	76
71	ACGR-M v3.0 Researcher Role	76
71.1	Owns	76
71.2	Required conduct	76
71.3	Prohibited	77
71.4	Outputs	77
72	ACGR-M v3.0 Role-Separated Hostile Reviewer Role	77
72.1	Review object	77
72.2	Mandatory procedure	78
72.3	Ruling table	78
72.4	Findings	79
72.5	Prohibited	79
72.6	Fairness	79
73	Professor Grumpy adversarial-AI project profile	79
73.1	Exact identification	79
73.2	Personality prompt	80
74	ACGR-M v3.0 Standards Index	80
75	QS-01 — Scientific Integrity, Evidence, and Literature	81
75.1	Required independent judgments	81
75.2	Claim and evidence gate	81
75.3	TCB, independence, and mutation gate	82
75.4	Literature and imported-theorem gate	82
75.5	Significance gate	82
75.6	Claim-surface consistency	83
75.7	Mechanical versus attested judgment	83
75.8	Non-waivable core	83
76	QS-02 — Computation, Operational Assurance, and Continuity	83
76.1	Cost applicability	83
76.2	Long-run authorization	84
76.3	Capability-safe operator surface	84
76.4	Live observability	84
76.5	Split closure	84
76.6	Platform/durability	84
76.7	Required fixtures	85

76.8	Non-waivable	85
77	QS-03 — Promotion, Role-Separated Hostile Review, and Successors	85
77.1	Lifecycle	85
77.2	Independence	85
77.3	Separate dimensions	86
77.4	Mandatory ruling and findings tables	86
77.5	Blocking predicate	86
77.6	Successors	86
77.7	Universal versus project profile	86
77.8	Required fixtures	87
78	QS-04 — Reviewer Artifact and Presentation Quality	87
78.1	Immutable review object	87
78.2	Reviewer time	87
78.3	Closed-world and non-mutation	87
78.4	Presentation QA	88
78.5	Identity	88
78.6	Required fixtures	88
79	QS-05 — External Reviewer and Public-Status Governance	88
79.1	Qualification authority	88
79.2	Reviewer record	88
79.3	Review quality	89
79.4	Aggregation	89
79.5	Public tiers	89
79.6	Appeals	90
79.7	Required fixtures	90
80	QS-06 — External Sources, Reproducibility, Application, and Measurement	90
80.1	Untrusted external-source ingestion	90
80.2	Reproducibility classes	90
80.3	Formalization and application bridge	91
80.4	Systems and empirical evidence scale	91
80.5	Environmental and energy claims	91
80.6	Non-waivable core	92
81	Version 2.7 Preservation and Clause Map	92
81.1	Section roots	92
81.2	Clause-level preservation invariant	101
82	Changelog from ACGR-M v2.7 to v3.0	101
82.1	Preserved foundation	101
82.2	Bonifications	101
82.3	Deliberately not changed	102
83	Migration from ACGR-M v2.7	102
83.1	New campaign	102
83.2	Existing campaign	102

83.3 Grandfathering	103
84 Summary	103

1 Core Principle

The compact form of ACGR-M remains:

Computation discovers. Counterexamples refine. Proof certifies.

Version 2.7 retains the operating principles introduced in Versions 2.1–2.2:

Do not optimize for accumulating successes. Optimize for verified mathematical information, and make each remaining failure maximally informative.

When progress stalls, do not merely search harder. Ask whether the failures show that the campaign is searching over the wrong mathematical objects.

Version 2.7 retains the long-campaign integrity principle:

An accepted claim is not an isolated file. It must retain an auditable dependency path to its assumptions, exact evidence, verifier semantics, and stated scope.

Version 2.7 also retains the consequence-sensitive and semantic-closure principles established in Versions 2.4–2.6:

Exactness is never tiered. Independent assurance is.

Assurance is justified by the failure mode it closes, not by the number of times a computation is repeated.

A generation may advance the research without advancing the theorem.

A valid object does not certify a claim until the verifier establishes the binding from the raw evidence, through the validated mathematical object, to the frozen claim actually being asserted.

Version 2.7 adds one further cold-start principle:

The full protocol and the campaign transfer brief are decision inputs. A compact template, status file, or summary is not a substitute when a fresh agent is making material research decisions.

The workflow is not “run a heuristic until something looks plausible.” It maintains hard boundaries between:

1. **Candidate discovery.** Search may be approximate, stochastic, learned, experimental, aggressively optimized, or generated by an AI system.
2. **Exact acceptance.** A candidate is accepted only when an exact mathematical oracle confirms the claim being made. Failure to discover a candidate is never a proof of nonexistence unless search completeness and pruning soundness are themselves certified.
3. **Residual harvesting.** Every unresolved case, failed construction, proof gap, boundary failure, mechanism plateau, and high-quality near-solution becomes structured evidence.

4. **Refinement.** The next mechanism is chosen to attack what the previous mechanism could not explain.
5. **Abstraction.** Finite evidence is periodically compressed into lemmas, invariants, parameterized families, reductions, classifications, or proof obligations.
6. **Proof.** Finite certificates prove finite claims. Universal theorems require arguments whose certified domain covers the full target domain.
7. **Proof attack.** A candidate proof is treated as a new object to falsify before it is trusted.

The discovery procedure may be fallible. The acceptance procedure may not be.

2 Version Lineage: What Version 2.3 Established and Version 2.7 Preserves

Versions 2.1 and 2.2 established the mature research and long-campaign integrity architecture: representation pressure, campaign-level autonomy, negative knowledge, scientific generation contracts, semantic regressions, verifier diversity, frozen benchmarks, forced recovery testing, claim/evidence dependency closure, controlled nondeterminism, transactional commits, schema-safe migrations, release-artifact closure, benchmark-contamination governance, and content-addressed evidence retention.

Version 2.3 is again conservative. It does not alter the correctness-first philosophy. It closes gaps that remain even when every file is preserved and every run is reproducible: shared verifier bugs, adaptive selection bias, unsound evidence compression, silent downstream damage after an oracle correction, and hidden dependence on a developer working state that cannot be reconstructed independently.

The new Version 2.3 additions are:

- an explicit **trusted computing base (TCB) and verifier-independence map**, so claims of independent verification disclose shared parsers, normalization code, libraries, schemas, arithmetic kernels, and assumptions that could create correlated failure;
- a formal distinction between **exploratory discovery/selection** and **confirmatory evaluation**, preventing a mechanism chosen after many adaptive trials from claiming unbiased improvement on the same evidence that selected it;
- **compression soundness and coverage obligations**: a compact rule set, lemma family, decision diagram, or summary remains a hypothesis until exact coverage/equivalence is established for any evidence it is intended to replace;
- a **verifier/oracle incident protocol** with claim quarantine, dependency-driven re-verification, preserved pre-correction semantics, and explicit repair provenance;
- **independent reconstruction and clean-room replication** for theorem-level, publication-grade, or major computational claims when feasible, using only the frozen handoff/release artifacts and documented dependencies rather than hidden developer state; and

- a stronger rule that **selection uncertainty and evaluation uncertainty are distinct**: stochastic dispersion, branch selection, hyperparameter search, and repeated benchmark exposure must not be collapsed into one favorable headline number.

The retained Version 2.2 foundations include:

- a stable **claim registry and evidence-dependency graph**, so accepted claims retain auditable links to assumptions, certificates, verifier semantics, scope, and downstream dependents;
- explicit **reproducibility classes** for bitwise-deterministic, seed-deterministic, semantically deterministic, and intentionally stochastic discovery mechanisms;
- **transactional checkpoint and worker-commit discipline**: isolated workers produce immutable result bundles and only the canonical controller commits verified state;
- mandatory **schema versioning and non-destructive migrations** for state, certificates, ledgers, and verifier-facing formats;
- a **release-artifact closure test** requiring the exact packaged bytes given to the human to be installed and tested from a clean reconstruction of the parent state;
- **benchmark-contamination governance**, so a repeatedly inspected held-out set cannot quietly remain labeled held-out after it influences solver design;
- **evidence retention and content-addressed compaction** rules for long campaigns; and
- an evidence-based **compute-escalation gate** after a plateau or representation barrier.

The retained Version 2.1 foundations include:

- a three-layer research model: **target claim**, **working hypothesis**, and **computational mechanism**;
- explicit optimization for **verified mathematical information per resource**, not raw throughput alone;
- mandatory **representation pressure** and periodic **compression/abstraction reviews**;
- **targeted challenge generation**, including adversarial residual generation when appropriate;
- distinction between **difficulty** and **novelty**;
- **frozen benchmark corpora**, held-out evidence, counterfactual evaluation, and deliberate ablations;
- a canonical **negative-knowledge registry** and semantic regression definitions;
- **verifier diversity**, mechanism-versus-campaign stops, append-only generation ledgers, immutable generation implementations, and whole-surface hash pinning;
- a canonical persistent-state manifest, forced interruption/recovery tests, external-frontier provenance, proof red-teaming, unattended autonomy, and scientific generation contracts;

- explicit numerical/objective discipline and certified completeness for computational negative/minimality claims;
- branch/merge provenance and competing-representation discipline; and
- conservative cross-campaign protocol evolution.

These are refinements of the architecture, not a change of purpose.

Versions 2.4–2.6 then strengthened the trust and operator path surrounding that architecture. Version 2.7 retains those additions in full:

- **consequence-sensitive assurance profiles A0–A3**: exact mathematical acceptance is invariant, while independent replication, audit, mutation testing, authenticity assurance, and publication scrutiny scale with consequence;
- **typed assurance outcomes** rather than one ambiguous PASS, including mathematical validity, raw-representation validity, claim-evidence binding, verifier regression status, package integrity, operator-surface status, verifier independence, authenticity, human audit, clean-room reconstruction, and frontier review;
- a strict **raw-evidence-before-normalization** rule, so coercion or canonicalization may simplify an already valid object but may never convert malformed evidence into something acceptable;
- explicit **claim-evidence binding**: a checker must establish that the validated object, parameters, domain, construction chain, scope, and exceptions prove the frozen claim actually being asserted;
- **adversarial mutation testing** for materially new verifier semantics, parsers, evidence formats, and claim-binding logic;
- a distinction among **integrity**, **provenance continuity**, and **authenticity**;
- **operator-surface closure**, release cleanliness, package/root ownership declarations, fail-closed wrappers, terminal-state sealing, bounded cancellation, long-run observability, and narrow operational-repair semantics;
- a formal separation between **generation release** and **mathematical result promotion**, including a cumulative append-only promoted-results portfolio;
- **skeptical human-audit dossiers** and evidence-minimal clean-room reconstruction for promoted computation-dependent claims;
- explicit separation of **developer/release**, **campaign/operator**, and **audit** roles so repeated runs answer distinct scientific questions rather than merely accumulating duplicate PASS labels;
- a **generation-transition brief** that tells the human what was established, what was not, how the generation compares with its parent, whether anything was promoted, the dominant residual, and the next attack before autonomous successor construction continues;

- a **single operator-transfer artifact** as the normal release path;
- a canonical claim/assurance record, applicability-closed assurance gates, compact reproducibility declarations, semantic migration rules, deterministic negative-result caching guidance, and an explicit protocol-complexity budget.

Version 2.7 makes one architectural correction to the presentation itself. The compact Version 2.6 document is no longer treated as sufficient onboarding for a protocol-naive agent. The detailed Version 2.3 material is restored as the primary normative teaching layer. The new material is integrated into that detailed structure instead of replacing it with a shorter summary.

Version 2.7 also promotes **agent-continuity transfer** to a protocol-level operational concern. The handoff package preserves scientific state; the new transfer ZIP preserves enough full-context guidance, relevant artifacts, and campaign rationale for a fresh agent instance to understand and use that state correctly. Section 45 defines the required semantics.

3 Normative Language

Version 3.0 retains the Version 2.7 normative vocabulary and uses the following terms deliberately:

- **MUST** means the rule is part of the correctness, provenance, or scientific-integrity contract. A generation that violates it is not an accepted ACGR-M generation.
- **SHOULD** means the rule is expected unless a problem-specific reason makes it inappropriate. The reason for omission must be recorded.
- **MAY** means the technique is optional and problem-dependent.

4 What the Agent Is Asked to Do

A minimal handoff instruction to another capable model is:

Solve the mathematical problem stated below using ACGR-M v3.0. Treat yourself as both the mathematician and the research engineer. Build a self-contained research campaign that searches for exact mathematical progress, verifies every accepted result exactly, preserves all prior evidence, converts failures into structured refinement data, periodically attempts abstraction and proof, and can be operated by a non-expert through a minimal safe interface.

Do not merely describe a strategy. Implement the research harness, verifier, persistent state, regression suite, diagnostics, provenance ledger, recovery tests, and handoff mechanism. Predeclare a scientific generation contract, distinguish navigation arithmetic from exact decision/acceptance arithmetic, and never infer nonexistence or minimality from incomplete search. At each generation, improve verified mathematical information gained per unit resource without weakening correctness, verification, regression coverage, certificate integrity, reproducibility, provenance, or proof quality.

Problem: [INSERT MATHEMATICAL PROBLEM HERE].

The agent should proceed autonomously unless a genuinely necessary external decision cannot be inferred.

The preceding paragraph is a bootstrap instruction, not a substitute for this document. Before making the first material design decision in a new or resumed campaign, a fresh agent **MUST** read the full governing protocol and, for an existing campaign, the current transfer brief and authoritative handoff/state artifacts.

5 Cold-Start Orientation for a Fresh or Context-Poor Agent

A recurring failure mode in AI-mediated research is not mathematical error but *context starvation*: a capable agent receives a compact protocol summary and a few status facts, then makes locally reasonable but globally poor choices because it does not know why earlier rules exist, what has already failed, which representations were deliberately rejected, or what the operator should and should not be asked to decide.

Version 2.7 therefore makes cold-start orientation explicit. A fresh agent entering an existing campaign **MUST** not design a successor mechanism until it can state, in its own working record, at least:

1. the frozen target claim and the exact scope currently under attack;
2. the acceptance oracle and the raw-evidence-to-claim binding that makes acceptance meaningful;
3. the current working hypothesis and computational mechanism, kept distinct from the target;
4. the strongest accepted and promoted claims, including their correctness, novelty/frontier status, and scope;
5. the active representation or representations and why they were chosen;
6. the dominant residual, plateau, or representation barrier exposed by the latest completed generation;
7. the relevant negative knowledge, including attractive ideas that should not be retried without new evidence;
8. the frozen benchmarks, evaluation roles, and any contamination history relevant to comparative claims;
9. the current operator, packaging, recovery, and environment constraints; and
10. the immediate next action expected from the agent and the human.

For a continuing campaign, the preferred orientation order is:

1. read the full governing ACGR-M protocol;
2. read the detailed transfer brief from the newest transfer ZIP;

3. inspect the newest authoritative handoff if one exists, verify project identity and hashes, and use it as the scientific source of truth for the completed run;
4. inspect the generation ledger, claim registry, negative-knowledge registry, and current release/authorization records as needed;
5. only then design, install, or materially alter the next generation.

If a transfer ZIP was created while a generation was still running, its live progress is contextual rather than final. A later completed handoff supersedes the transfer snapshot for canonical scientific results. The transfer brief remains useful for campaign rationale, operator constraints, and continuity decisions that a machine handoff may not express pedagogically.

A new agent should continue the campaign, not rediscover the campaign.

6 Suitability Check Before Search

Before implementing a solver, the agent **MUST** determine whether the problem is suitable for ACGR-M.

A problem is especially suitable when at least one of the following exists:

- an exact finite witness that can be checked mechanically;
- a decidable finite subproblem whose failures are informative;
- a symbolic identity, algebraic condition, SAT/SMT formulation, integer program, or theorem-prover obligation;
- a family of constructions with explicit coverage domains;
- a falsifiable intermediate conjecture;
- a structured search space in which near-solutions reveal useful residual information;
- a natural notion of minimal obstruction, extremal example, hard residue, or boundary case;
- multiple mathematically distinct representations that can cross-check one another.

If no reliable correctness oracle can be defined, the first research objective is the oracle. A solver that cannot tell success from failure is not ready for ACGR-M.

7 Three-Layer Research Model

At all times the project **MUST** distinguish three levels.

1. Target claim

The actual theorem, conjecture, construction objective, classification, bound, or finite target the campaign is trying to establish or refute.

2. Working hypothesis

A narrower structural statement whose truth would materially advance the target claim. Examples include a bounded parameter hypothesis, a covering-family hypothesis, a symmetry reduction, a conjectured lemma, or a finite classification.

3. Computational mechanism

The current algorithm, search grammar, representation, move set, proof tactic family, solver encoding, or certificate generator used to test the working hypothesis.

A failure at one level must not be promoted to another. In particular:

Failure of the current mechanism is not automatically failure of the working hypothesis, and failure of the working hypothesis is not automatically refutation of the target claim.

Every handoff **MUST** say explicitly which layer a failure belongs to.

8 Mathematical and Campaign State

At generation g , the persistent mathematical state should be modeled as

$$S_g = (T, H, R, \Theta, Q, C, F, N, G, B, D, P),$$

where:

- T is the frozen target claim;
- H is the set of working hypotheses and their statuses;
- R is the active mathematical representation or representations;
- Θ is accepted theory: definitions, lemmas, reductions, constructions, proof fragments, and dependencies;
- Q is the claim registry and auditable evidence-dependency graph;
- C is the certificate/witness store;
- F is the structured residual and counterexample store;
- N is negative knowledge: failed mechanisms, dead ends, and retry conditions;
- G is the regression suite and its semantic equivalence definitions;
- B is the frozen benchmark and held-out evidence corpus;
- D is diagnostics and performance data;
- P is provenance: generation lineage, hashes, environment, external frontier, and runtime records.

The wider campaign state may contain multiple research lanes, but every lane must point to the same frozen target claim or explicitly declare a branch target.

Every mathematical claim **MUST** have a status such as:

Status	Meaning
Hypothesis	Proposed but not established.
Empirically supported	Survived finite tests only.
Computationally certified	An exact finite certificate establishes the stated finite claim.
Proof candidate	A deductive argument or checked object appears complete but has not yet passed the required proof red-team/independent audit.
Formally proved	A deductive proof or checked proof object has passed the campaign’s theorem-level acceptance and red-team requirements.
Quarantined	Previously accepted or candidate evidence is temporarily non-reliable because an upstream verifier, oracle, schema, assumption, or dependency incident is unresolved.
Refuted	An explicit exact counterexample exists.
Obsolete	Superseded, corrected, or intentionally abandoned.

No finite experiment may silently be promoted to a universal theorem.

9 Claim Characterization and Canonical Claim/Assurance Record

Every significant human-facing claim **SHOULD** be characterized independently along three axes:

Correctness		Novelty / Frontier Status		Scope
-------------	--	---------------------------	--	-------

Correctness uses the campaign claim state (for example Hypothesis, Empirically supported, Computationally certified, Proof candidate, Formally proved, Quarantined, Refuted, or Obsolete). **Novelty/frontier status** states what the current literature review supports, with values such as Known, Apparently novel, Novelty uncertain, or Not reviewed. “Apparently novel” means that no prior explicit result was found under the recorded review; it is not an absolute proof of historical priority. **Scope** states what the claim actually covers: a single finite instance, a finite enumerated domain, a parameterized family, a representation-level statement, a reduction/classification, or the universal target theorem.

A claim may be exactly correct yet already known; genuinely new yet narrow; or computationally important without resolving the target. Reports **MUST** not blur these axes.

Each important accepted or promoted claim **SHOULD** also have one authoritative machine-readable record from which human status, promotion status, audit headers, and summary assurance are rendered. The protocol does not require one physical filename; it requires one semantic source of truth. A minimal record should bind:

- stable claim identifier and canonical statement hash;
- correctness, novelty/frontier status, and scope;

- assurance profile and typed assurance components;
- assumptions and upstream claim identifiers;
- evidence root and verifier/checker root;
- TCB/independence root;
- frontier-review identity and date;
- promotion status; and
- generation last reverified.

Human-readable reports should be derived from this canonical record or from an equivalent canonical registry rather than being manually maintained as a second source of truth.

10 Claim Registry and Evidence Dependency Closure

A long campaign accumulates claims faster than a human can safely remember which assumptions, certificates, finite domains, or verifier versions support each one. File preservation alone is therefore insufficient.

Every nontrivial accepted claim, major working hypothesis, completeness claim, and proof-relevant intermediate result **MUST** have a stable identifier in a claim registry. The registry entry should record:

- the exact statement, or a canonical statement hash plus a human-readable rendering;
- status and scope, including parameter ranges and exceptional cases;
- explicit assumptions;
- upstream claim identifiers;
- certificate, witness, dataset, or proof-object hashes supporting the claim;
- the exact verifier or checker identity and semantic version used for acceptance;
- the generation in which the claim was introduced and last reverified;
- downstream claims known to depend on it;
- supersession, correction, or obsolescence events.

The registry need not be a full formal proof assistant. It **MUST**, however, be sufficient to answer:

What exact evidence supports this claim, under which assumptions and verifier semantics, and what downstream conclusions become suspect if one dependency changes?

If an upstream assumption, certificate semantics, finite-domain definition, claim-evidence mapping, or acceptance oracle changes materially, dependent claims must not remain silently green. They **MUST** be marked for re-verification or superseded until their dependency closure is re-established.

For theorem-level, completeness, minimality, and campaign-frontier claims, the handoff **MUST** include the full active dependency closure or an independently verifiable content-addressed representation of it.

A recommended machine-readable artifact is `claim_registry.jsonl` or an equivalent append-only claim/evidence graph.

11 Claim-Bound Exact Acceptance

Discovery and acceptance **MUST** remain independently designed. Approximate arithmetic, heuristics, learned scores, and stochastic search may guide discovery; they may not replace the exact route that establishes an accepted mathematical claim.

Validate raw evidence before normalization

For any accepted certificate, witness, matrix, graph, integer tuple, serialized proof object, or other structured evidence, the verifier **MUST** validate the *original representation* before any conversion that could turn an invalid object into a valid-looking one.

Normalization may simplify a valid object; it may never be used to decide whether the original object was valid.

Examples of raw invalidity that must be rejected before coercion when relevant include out-of-domain numeric values, floating values where exact integers are required, Booleans accepted as integers, strings accepted as numbers without an explicitly valid schema, incompatible signedness or data type, malformed dimensions, missing or duplicated components, invalid encodings, and inconsistent labels or metadata. Derived metadata is untrusted input when it can be recomputed.

Bind the validated object to the frozen claim

Version 2.7 retains the Version 2.6 semantic-closure requirement that an acceptance route establish the full relation

$$\text{raw evidence} \implies \text{validated mathematical object} \implies \text{frozen claim}.$$

It is not enough to validate an object in isolation. Where relevant, the verifier must also establish:

- parameter and finite-domain identity;
- dimensions, orders, labels, orientation, and boundary conditions;
- the exact transformation or construction chain from witness to asserted result;
- scope restrictions and exceptional cases;
- consistency between the frozen claim statement and recomputed metadata; and

- the versioned semantics of every reduction or equivalence used to connect evidence to the claim.

A certificate valid for claim C cannot silently certify a broader, differently parameterized, differently labeled, or differently scoped claim C' . Broadening scope or changing the claim-evidence mapping requires explicit re-verification.

Arithmetic roles

Campaigns **MUST** distinguish:

1. **navigation arithmetic**: approximate, heuristic, learned, spectral, probabilistic, or floating quantities used to decide where to search;
2. **decision arithmetic**: quantities used for checkpoints, best-state comparisons, pruning, branch decisions, or claims of improvement;
3. **acceptance arithmetic**: the exact route that establishes the mathematical claim.

Decision arithmetic deserves special care because an error there can alter which exact obligations are ever presented to the acceptance oracle. Section 24 gives the detailed numerical discipline retained from Version 2.3.

12 Trusted Computing Base and Verifier Independence

“Independent verification” is meaningful only to the extent that the verification routes do not share the same unexamined failure source. Two checkers that call the same parser, normalization routine, arithmetic kernel, schema decoder, or domain-reduction code may fail identically while appearing independent.

For every important acceptance route, the campaign **SHOULD** maintain a compact **trusted computing base (TCB) manifest**. For theorem-level, completeness/minimality, and major computational claims, it **MUST** identify at least:

- the exact verifier/checker executable or proof object checker;
- parsers, canonicalizers, domain reducers, certificate decoders, and schema versions on which acceptance depends;
- arithmetic, SAT/SMT, proof-assistant, numerical, or external solver components whose correctness is trusted rather than re-established by the certificate;
- shared code or shared mathematical reductions between purportedly independent verification paths;
- which assumptions are checked mechanically and which remain human- or literature-trusted.

Verifier diversity **MUST** therefore be reported with an **independence disclosure**. A second checker may still be valuable if it shares components with the first, but the campaign must not describe it as fully independent without stating the common trusted base.

The exact acceptance checker **SHOULD** minimize its TCB where practical: prefer a small standalone verifier, proof-producing solver, proof assistant kernel, or independently auditable exact checker over a large discovery stack. This is not a requirement to re-prove the operating system or hardware. It is a requirement to make the mathematically relevant trust boundary visible.

A recommended artifact is `tcb_manifest.json` together with `verifier_independence.json`.

The independence disclosure must state whether purportedly separate routes share the same parser, coercion rules, canonicalizer, claim-binding function, schema decoder, or derived-metadata logic. Two executables that accept through the same permissive normalization or claim-binding layer are not strongly independent even if their outer code differs.

13 Consequence-Sensitive Assurance and Gate Closure

Exact mathematical acceptance is invariant. What scales with consequence is the amount of independent reconstruction, audit, mutation testing, operator-path validation, authenticity assurance, and publication-grade scrutiny surrounding the claim.

Exactness is never tiered. Independent assurance is.

Profile	Purpose	Minimum surrounding assurance
A0 – Exploratory	Mechanism learning, residual discovery, representation search	Exact local acceptance for any accepted object, protected state, provenance, relevant regressions, and honest exploratory labeling. No promoted mathematical claim is implied.
A1 – Generation Release	Safely transfer an authorized generation to the operator	A0 plus release manifest, exact packaged-byte closure, operator-surface validation for declared platforms, forced recovery, release cleanliness, ownership checks, and typed assurance summary.
A2 – Promoted Computational Claim	Promote a finite theorem, exact classification, completeness/minimality result, construction, reduction, or comparable computation-dependent result	A1 plus claim-promotion gate, exact coverage where applicable, independent checker, skeptical human-audit dossier, clean-room reconstruction unless redundant, TCB/independence disclosure, and current frontier review.
A3 – Universal / Major Priority Claim	Universal theorem, publication-grade major claim, or priority-bearing result	A2 plus proof red-team, strongest feasible independent/formal route, renewed frontier review, publication/novelty gate, and authenticity anchoring where provenance itself matters.

A lower profile may never be used to weaken mathematical correctness.

Minimal sufficient assurance

For every expensive verification, reconstruction, replication, or audit step above the exact acceptance oracle, the campaign **SHOULD** name the failure mode it is intended to detect. Useful classes

include:

- **F1 – acceptance semantics:** invalid mathematics is accepted;
- **F2 – shared implementation/TCB:** independent-looking routes share the same hidden bug;
- **F3 – completeness/coverage:** admissible cases are omitted;
- **F4 – packaging/operator path:** tested bytes differ from what the human executes, or failures propagate incorrectly;
- **F5 – provenance/authenticity:** artifact lineage or identity is weaker than claimed;
- **F6 – scientific interpretation:** a correct computation is overstated in novelty, scope, or theorem consequence.

Repeating the same computation through the same parser, TCB, data, and acceptance path does not become materially independent merely because it ran twice. At A1, representative real scientific work is normally sufficient for release closure; a full duplicate campaign requires a stated assurance purpose.

Typed assurance outcomes

At A1 and above, relevant assurance dimensions **MUST** be reported separately. A recommended rendering is:

```
MATHEMATICAL_VALIDITY ..... PASS / FAIL / N-A
RAW_REPRESENTATION_VALIDITY .. PASS / FAIL / N-A
CLAIM_EVIDENCE_BINDING ..... PASS / FAIL / N-A
VERIFIER_REGRESSIONS ..... PASS / FAIL / N-A
PACKAGE_INTEGRITY ..... PASS / FAIL / N-A
OPERATOR_SURFACE ..... PASS / PARTIAL / FAIL
VERIFIER_INDEPENDENCE ..... STRONG / PARTIAL / WEAK / UNKNOWN
ARTIFACT_AUTHENTICITY ..... VERIFIED / UNVERIFIED / N-A
HUMAN_AUDIT ..... PASS / FAIL / NOT_REQUIRED
CLEAN_ROOM_RECONSTRUCTION .... PASS / FAIL / NOT_REQUIRED
FRONTIER_REVIEW ..... CURRENT / STALE / NOT_REQUIRED
```

A wrapper or presentation-layer failure may never print an overall PASS that masks a failed component.

Applicability closure

A gate is **applicability-closed**: it may report PASS only when every component required by the active assurance profile is either PASS or explicitly NOT_REQUIRED/N-A with a recorded reason. PARTIAL, UNKNOWN, STALE, or an unexecuted required check is not authorization. Partial results remain useful diagnostics; they do not silently satisfy a stronger gate.

14 Counterexamples, Residuals, and Negative Knowledge

A “counterexample” in ACGR-M is broader than a direct disproof. A development batch may contain:

- a direct counterexample to a conjectured lemma;
- a construction failure on a target instance;
- a coverage gap where no known theorem applies;
- an abstention by the current solver;
- an unclosed proof obligation;
- a boundary-condition or composition failure;
- hidden assumption leakage;
- a case that defeats the current certificate grammar but not the target theorem;
- a high-quality near-solution that exposes the geometry of the residual;
- a mechanism plateau in which repeated search returns to the same basin;
- a performance barrier caused by the chosen representation.

For each residual, the important object is not only “what failed,” but *why the current mechanism appears to fail*.

Whenever affordable, a surviving residual **SHOULD** carry a positive exact witness explaining why it survives the current mechanism: a minimal obstruction, violated proof obligation, uncovered residue, canonical near-solution, or other explicit object.

Failed mechanisms **MUST** be recorded in a negative-knowledge registry containing at least:

- what was tried;
- on what evidence and parameter range;
- whether failure was mathematical, representational, or engineering;
- what was learned;
- whether the idea is abandoned, deprioritized, or merely unproven;
- the condition under which it may be retried.

A recommended field is `retry_only_if`, preventing later generations from rediscovering attractive dead ends without new evidence.

When an exhaustive deterministic mechanism has already established that a particular exact state/configuration fails, later generations **SHOULD** reuse that negative result rather than repeat the same low-information computation, provided the cache key includes every mathematically relevant input, mechanism/configuration version, verifier semantics, and claim-binding version. Cache invalidation must be explicit whenever those semantics change.

15 The Mature ACGR-M Loop

For generation $g = 0, 1, 2, \dots$:

1. **Freeze the target claim.** Write the exact theorem, construction objective, or finite target set.
2. **Freeze the acceptance oracle.** Define the exact condition separating valid from invalid results.
3. **Snapshot the external frontier.** Record the mathematical results, datasets, solver versions, and assumptions that define what is already known.
4. **Reproduce the baseline.** Recheck inherited certificates, regressions, benchmark results, and the strongest known state.
5. **State the working hypothesis and representation.** Say what structural claim is being attacked and what mathematical objects encode it.
6. **Run bounded discovery work.** Search may use exhaustive methods, local search, algebraic constructions, SAT/SMT, integer programming, theorem proving, symbolic manipulation, stochastic search, learned heuristics, or hybrids.
7. **Accept only exact, claim-bound results.** Approximate objectives may navigate discovery but cannot establish accepted mathematics. Validate raw evidence first, then establish the binding from the validated mathematical object to the frozen claim.
8. **Classify and preserve residuals.** Store failures, near-solutions, proof gaps, trajectories, diagnostics, and positive survival witnesses.
9. **Learn reusable structure.** Convert successful certificates or failure patterns into exact families, rules, invariants, clauses, lemmas, or filters where justified.
10. **Construct targeted challenges.** Once ordinary cases are low-information, preferentially generate examples that evade what is already understood. Use explicitly adversarial generation when the problem admits it; otherwise use the strongest justified falsification-oriented challenge generator.
11. **Measure information gain.** Evaluate useful mathematical progress per unit resource, not raw throughput alone, and distinguish exploratory selection evidence from confirmatory evaluation.
12. **Compress.** Attempt to replace accumulated finite evidence with a smaller number of structural explanations.
13. **Certify compression scope.** If a compressed artifact will replace evidence or justify pruning, establish its soundness/coverage/equivalence exactly; otherwise retain it as a working hypothesis.
14. **Challenge the representation.** Ask whether the current search space is still the right mathematical object.

15. **Attempt proof.** When compression suggests a universal mechanism, formulate and prove the strongest justified statement.
16. **Attack the proof.** Red-team every lemma, domain boundary, quantifier, exception, hidden assumption, and computational premise.
17. **Independently reconstruct major claims.** For theorem-level or major computational claims, verify that the frozen scientific artifacts reproduce without hidden developer state, using a distinct audit path when feasible.
18. **Decide result promotion separately from generation success.** A productive generation may add important negative knowledge or a new mechanism without producing a promotable mathematical result.
19. **Prepare the generation transition brief.** State what was established, what was not, how the generation compares with its parent, the promotion decision, the dominant residual, and the next attack.
20. **Decide mechanism continuation, lane transition, or generation handoff.** Use predefined evidence-based criteria.
21. **Optimize under a hard quality constraint.** Improve speed, memory, selectivity, and autonomy only if mathematical quality is unchanged or stronger.

The loop is therefore not simply “search, fail, search harder.” It is a controlled process for converting computation into increasingly compressed mathematical structure.

16 Scientific Generation Contract

Before a generation is authorized to consume substantial resources, it **MUST** have a short, machine-readable and human-readable experimental contract. The purpose is not bureaucracy; it is causal interpretability.

The contract **MUST** state:

- the inherited frontier and evidence on which the generation is based;
- the dominant failure mechanism or unresolved question being attacked;
- the working hypothesis and active representation;
- the intervention: what mathematical, algorithmic, representational, verification, or engineering change is being introduced;
- the expected causal mechanism by which that change should help;
- the frozen benchmark or baseline against which the change will be measured;
- which evidence is exploratory and, where meaningful, which untouched or renewed evidence will be reserved for confirmatory evaluation;

- the primary mathematical-progress metrics and secondary compute metrics;
- the stop, lane-transition, and handoff criteria;
- the exact quality gates that may not regress;
- any planned ablations needed to separate multiple simultaneous changes.

A generation is allowed to falsify its own hypothesis. That is a valid scientific outcome. It must not be judged a failure merely because it produces no new target certificate if it establishes a new boundary, eliminates a plausible mechanism, improves verification, or reveals a better representation.

Post hoc metrics may be reported, but they must be labeled post hoc. A run must not retroactively redefine success merely because an unexpected statistic looks favorable.

17 Representation Pressure

The strongest improvements in a difficult campaign often come from changing the mathematical object being manipulated rather than making the existing loop marginally faster.

At every handoff, the agent **MUST** ask:

What object has become the dominant cost, and is that object one abstraction level too low?

Examples of representation changes include:

- individual instances $\rightarrow$ parameterized families;
- objects $\rightarrow$ symmetry classes or quotient objects;
- raw solutions $\rightarrow$ minimal obstructions or antichains;
- explicit enumerations $\rightarrow$ generating functions or decision diagrams;
- repeated local states $\rightarrow$ sufficient statistics or shared intermediate states;
- numeric search $\rightarrow$ exact symbolic recurrences;
- isolated certificates $\rightarrow$ coverage theorems;
- one coordinate system $\rightarrow$ a dual, modular, geometric, algebraic, graph-theoretic, or proof-theoretic representation.

If progress is only incremental across multiple refinements, the next generation **SHOULD** include at least one representation-level attack rather than merely increasing runtime.

When feasible, the campaign **MAY** maintain multiple competing representations. Convergence between independent representations is especially valuable evidence; disagreement is diagnostic.

18 Discovery, Compression, and Proof

Finite search can become extremely powerful while remaining infinitely far from a universal theorem. Therefore ACGR-M requires deliberate abstraction pressure.

The intended trajectory is:

finite witness $\rightarrow$ structural pattern $\rightarrow$ lemma $\rightarrow$ general construction or reduction $\rightarrow$ proof.

A compression review **MUST** occur at every generation handoff and **SHOULD** also occur when any of the following is observed:

- thousands of successful certificates cluster into a small number of forms;
- the adversary increasingly produces variants of already-understood mechanisms;
- learned rules grow rapidly while genuinely novel residuals become rare;
- the dominant cost becomes maintaining the rule/certificate set rather than discovering new structure;
- a finite covering argument appears close to complete;
- independent representations converge on the same invariant.

The compression review asks, for example:

Can 10,000 rules be generated by 100? Can 100 be generated by 5 lemmas? Can those lemmas cover the entire residual domain? Can a proof obligation replace an open-ended search?

Failure to compress is itself evidence. The reason compression failed becomes input to the next refinement.

Repeated saturation of the same exact structural boundary is itself a compression signal. If increasingly adversarial generations repeatedly reach the same boundary, produce variants of the same certificate mechanisms, and fail to generate a higher obstruction, the next generation **SHOULD** allocate a substantial lane to explaining that boundary through lemmas, coverage arguments, alternative representations, or proof obligations rather than merely extending the same search.

19 Compression Soundness and Coverage

Compression is a research objective, not an automatic proof step. A concise explanation can be elegant, predictive, and still fail on an exceptional case.

Whenever a compressed artifact—for example a lemma family, rule grammar, decision diagram, generating function, classification, symmetry quotient, or canonical representative set—is used to *replace* a larger body of evidence or to justify pruning, the campaign **MUST** state the exact relation being claimed:

- soundness only (everything generated/accepted by the compression is valid);

- completeness/coverage (every object in the stated domain is represented or covered);
- equivalence (both directions hold under a declared semantic relation);
- heuristic summary only (useful for discovery but not a substitute for the underlying evidence).

A compression may be promoted from heuristic summary to evidence-preserving replacement only after the required soundness/coverage/equivalence property is established exactly or by a separately certified completeness argument. Until then, the underlying irreducible evidence remains part of the active dependency closure and must not be discarded merely because the compressed model fits all observed cases.

Compression failures and exceptional uncovered cases are priority residuals. They are often more informative than another successful compression example.

20 Targeted Challenge Generation and Novelty

After the easy population is understood, random or consecutive sampling may become mathematically inefficient. A million routine successes may contain less information than a handful of cases deliberately constructed to evade all known mechanisms.

Therefore, once ordinary sampling is low-information, the solver **SHOULD** use targeted challenge generation. When exact known structure permits it, this may take the stronger form of adversarial residual generation: construct cases that satisfy as many known success conditions as possible while avoiding their coverage. When adversarial construction is unnatural or unjustified, use problem-appropriate boundary, held-out, metamorphic, proof-obligation, or falsification-oriented challenges instead.

A challenge generator must never change the correctness oracle. It changes only which cases receive expensive attention.

Difficulty and novelty are different:

- **Difficulty** measures how much of the current mechanism a case defeats: larger proven-minimal parameters, deeper proof search, more failed rules, larger residual, and so on.
- **Novelty** measures whether the case exposes a structurally new failure mechanism rather than another member of an existing cluster.

Novelty scoring is problem-dependent and **MAY** be heuristic. It must never be confused with proof strength. Possible signatures include residue patterns, certificate forms, factorization structure, proof traces, move sequences, symbolic normal forms, or graph invariants.

When reporting “hardest” cases, the system **MUST** state whether hardness is *proved minimal* or only “first found under the current search order.” A heuristic first-success metric must never be labeled minimal.

21 Adaptive Learning and Counterfactual Utility

An adaptive solver may learn exact rules online and use them to filter future work. This is encouraged when correctness is preserved, but growth in the rule count is not itself evidence of

improvement.

Every learned rule **SHOULD** record:

- the exact source certificate or derivation;
- the time/generation of discovery;
- the domain it covers;
- whether it is logically redundant with existing rules;
- estimated or replay-measured work eliminated;
- cost of storing and applying it;
- whether it improves hard-case density or only raw rejection.

When practical, adaptive learning **SHOULD** be evaluated counterfactually by replaying a held-out stream or benchmark with and without the learned mechanism. This distinguishes genuine intelligence gain from simply adding filters until almost nothing survives.

22 Exact Acceptance and Verifier Diversity

Discovery and acceptance **MUST** remain independently designed. Section 11 is normative: raw representation is validated before coercion, and acceptance must establish the binding from evidence to the exact frozen claim.

The search algorithm may be complicated. The exact verifier should be as small, deterministic, and auditable as practical.

For important new mechanisms, generation transitions, and final claims, ACGR-M **SHOULD** prefer **verifier diversity**: a materially different mathematical checking route rather than merely rerunning the same implementation. The report must distinguish genuinely different checking logic from routes that share a substantial trusted computing base.

Examples include:

- optimized search result checked by literal brute force on the relevant finite domain;
- numeric discovery checked by exact symbolic identity;
- custom solver checked by an independent SAT/SMT encoding;
- constructive certificate checked by a short standalone verifier;
- theorem candidate checked by a proof assistant or proof-producing solver;
- two independently implemented exact enumerators compared under the correct mathematical equivalence relation.

Routine millions-scale certificates need not all be checked twice by expensive independent systems if a simple immutable exact oracle already checks every result. However, novel mechanism classes and release gates should receive stronger independent audits.

Adversarial verifier mutation testing

For a materially new verifier, parser, evidence schema, or claim-binding route used at A1 or above, release QA **MUST** include adversarial mutation tests appropriate to the semantics. Begin with valid evidence, mutate it into invalid evidence or a false claim binding, and verify rejection.

When relevant, the suite should include type confusion, illegal values, malformed dimensions, meta-data relabeling, truncation, duplicated or missing components, boundary values, stale derived fields, claim-scope changes, invalid encodings, and altered evidence accompanied by recomputed internal checksums or manifests. The last class is particularly important: a self-consistent package with freshly recomputed hashes must still be rejected if its mathematics or claim binding is false.

23 Verifier/Oracle Incidents, Quarantine, and Repair

The exact oracle is a scientific dependency. If a bug, semantic ambiguity, unsound pruning rule, parser error, certificate-decoding error, claim-binding error, domain mismatch, or materially incorrect external assumption is discovered, the campaign **MUST** treat it as an incident rather than silently patching the checker.

The response **MUST** include:

1. freeze and hash the incident-discovered state and affected verifier semantics;
2. identify the earliest affected generation and the dependency-graph region that may rely on the faulty component;
3. mark affected claims and certificates **Quarantined**; they may not support downstream acceptance while the incident is unresolved;
4. preserve the old verifier and evidence so the failure can be reproduced;
5. implement the correction as a new audited verifier/generation or schema event, not an invisible rewrite;
6. re-verify the affected dependency closure under the corrected semantics;
7. append an incident report stating which claims were restored, weakened, refuted, superseded, or remain unresolved;
8. rerun release-closure, regression, and relevant benchmark checks before reauthorization.

An oracle correction may invalidate prior mathematical claims; preserving historical files is not enough if their status remains falsely green. Quarantine is the mechanism that prevents a known-bad dependency from continuing to certify downstream work.

A recommended artifact is `incident_log.jsonl` with stable incident identifiers referenced by the claim registry and generation ledger.

24 Numerical and Objective Discipline

Approximate arithmetic may guide discovery, but numerical convenience must never silently alter mathematical semantics.

The campaign **MUST** distinguish at least three roles:

1. **Navigation arithmetic:** approximate, floating-point, spectral, probabilistic, learned, or heuristic quantities used to choose where to search.
2. **Decision arithmetic:** quantities used for checkpointing, best-state comparisons, branch decisions, pruning, or claims of improvement.
3. **Acceptance arithmetic:** the exact oracle that establishes the mathematical claim.

When the underlying mathematical state or objective is discrete, the agent **SHOULD** actively investigate exact incremental maintenance, canonical integer/rational representations, or exact re-computation at important decision boundaries. If a fast approximate engine is retained, mathematically integral or rational quantities should be canonicalized only when justified and independently audited exactly before consequential decisions.

Cosmetic rounding must never change search state, pruning, acceptance, or certificate contents. Tolerances must have a mathematical or numerical-analysis justification and be recorded.

A heuristic objective is a navigation device, not automatically a measure of “distance to proof.” A smaller residual, energy, loss, or gap may be useful operationally while having no theorem-level interpretation. Reports must distinguish objective improvement from strengthened mathematical knowledge.

25 Regression Protection and Semantic Equivalence

Every previously accepted theorem, certificate, invariant, critical boundary case, and known counterexample must remain protected.

A regression test **MUST** specify what equality means. Depending on the object, acceptable equivalence may be:

- exact ordered-stream equality;
- canonical set equality;
- multiset equality;
- certificate-equivalent output;
- logically equivalent theorem coverage;
- equal objective value plus independently certified feasibility;
- another explicitly defined mathematical equivalence.

Incidental iteration order must not reject a mathematically equivalent generation unless deterministic ordering is part of the reproducibility or hashing contract. If order matters operationally, canonicalize it explicitly.

26 Reproducibility Classes and Controlled Nondeterminism

Reproducibility is not one binary property. A campaign may use stochastic search, parallel scheduling, external solvers, or asynchronous workers while still maintaining exact mathematical acceptance.

Each substantial mechanism **MUST** declare the strongest reproducibility class it actually satisfies:

1. **Bitwise deterministic.** Identical authorized inputs and environment produce identical bytes.
2. **Seed deterministic.** A recorded random seed stream and fixed environment reproduce the same trajectory and outputs.
3. **Semantically deterministic.** Scheduling, ordering, or low-level execution may differ, but the canonical mathematical output is identical under a declared semantic equivalence relation.
4. **Stochastic discovery with exact acceptance.** Search trajectories may legitimately differ; every accepted certificate remains deterministic and independently checkable, and statistical performance claims are evaluated with an explicit sampling protocol.

For non-bitwise mechanisms, the provenance record **SHOULD** include the random-number generator and version, seed derivation rule, worker/process count, external solver version, and any scheduling or hardware feature known to affect the discovery trajectory.

Nondeterminism may affect *which witness is found first*; it must never silently affect the truth semantics of exact acceptance.

A computational completeness, nonexistence, or minimality claim **MUST** not depend on schedule-dependent omission, opportunistic work stealing, or stochastic coverage unless the completeness argument independently certifies that every required case was covered.

Benchmarks of stochastic or schedule-sensitive mechanisms should report repeated-run dispersion or another justified uncertainty summary rather than presenting one favorable run as a deterministic speed claim.

27 Finite Exhaustion, Minimality, and Negative Claims

Computational evidence can establish nonexistence, impossibility, exhaustiveness, or minimality only when the campaign certifies the completeness of the search or deduction that supports the negative claim.

For such a claim, the project **MUST** preserve enough evidence to establish:

- the exact finite domain or logically exhaustive case split;
- why every admissible case is covered;

- the soundness of every pruning rule, symmetry reduction, dominance rule, or learned exclusion used to remove cases;
- the exact result for every surviving case or a proof-producing equivalent;
- the semantics of any minimality ordering;
- the version and hash of the checker that validates the completeness certificate.

A timed-out, plateaued, or very large search is evidence about the mechanism, not evidence of mathematical nonexistence. Likewise, “best found,” “first found,” and “smallest observed” must never be reported as “minimal” without a completeness argument.

28 Generations, Mechanisms, and Campaigns

A **generation** is an immutable, versioned research implementation with a frozen executable surface, assumptions, parameters, and acceptance logic.

A **mechanism** is one search lane, representation, tactic family, move set, construction class, or proof attack within a generation.

A **campaign** is the full effort on the frozen target claim and may span many generations and mechanisms.

This distinction matters because:

A mechanism stop does not necessarily imply a campaign stop or even a generation handoff.

A mechanism may be scientifically exhausted while other prevalidated lanes remain productive. A generation may therefore contain a queue of approved lanes, provided:

- each lane has a stated hypothesis, budget, and stop condition;
- the controller can invalidate later lanes if new evidence breaks their assumptions;
- lane transitions are logged;
- no lane bypasses exact acceptance or regression gates;
- the generation hands off when no worthwhile approved lane remains, a generation-level criterion is met, or the operator requests a stop.

Terminal-state sealing

A terminal generation state **MUST** be non-resumable through the ordinary run launcher. Once a predefined terminal condition, boundary stop, representation-review budget, exact target event, or other generation-level terminal reason has been durably committed, continuation requires an explicit successor generation, branch, or separately authorized continuation event whose provenance is recorded. Terminal, quarantine, verifier-failure, corruption, and safety-stop events must bypass ordinary heartbeat throttling and be printed/flushed immediately.

Accidental post-terminal work is preserved as an incident or separately identified continuation; it is never silently folded back into the intended terminal checkpoint.

Operational repair events

Not every operator-path defect requires a new mathematical generation, but an in-place repair must not blur provenance. A change may be classified as a narrow **Operational Repair Event** only if it provably does not alter discovery semantics, acceptance semantics, claim-evidence binding, mathematical evidence, finite-domain meaning, or executable scientific logic producing accepted results.

A safe operational repair **MUST** identify the exact project and authorized generation; verify expected hashes, ownership, lock/liveness state, and parent identity; declare precisely which wrapper or operational file may change; preserve before/after evidence; fail closed if authorization is ambiguous; append a repair event; and rerun the affected operator-surface tests and relevant regressions. Otherwise the change is a new generation, migration, or verifier incident as appropriate.

29 Stop Signals and Handoff Criteria

Stop and checkpoint signals include:

1. **Exact target success.** A target theorem, witness, construction, or counterexample passes exact acceptance.
2. **Working-hypothesis boundary event.** A case reaches or breaks a predefined structural threshold worth immediate analysis.
3. **Diagnostic threshold.** A residual, energy, proof gap, coverage gap, or novelty threshold becomes informative enough to justify refinement.
4. **Mechanism plateau.** No meaningful improvement occurs under the mechanism’s predefined criterion.
5. **Mechanism failure.** Diagnostics show repeated return to the same basin or a structural barrier.
6. **Representation barrier.** The dominant cost has moved to an object the current representation handles poorly.
7. **Compression opportunity.** Accumulated evidence appears ready for lemma/proof extraction.
8. **Resource budget.** The planned active runtime, CPU/GPU budget, memory budget, or search budget is reached.
9. **Campaign queue exhausted.** No prevalidated productive lane remains.
10. **External stop.** The operator requests a clean stop.

Stop/handoff criteria **SHOULD** be declared before the run begins. They may be revised only with a logged justification; they should not be retrofitted merely to make a run look successful.

30 Benchmarking and Information Efficiency

Raw speed is not enough. A generation can process fewer final candidates per second while being dramatically better if it filters understood cases earlier and concentrates expensive work on informative residuals.

Each campaign **SHOULD** maintain frozen benchmark corpora such as:

- ordinary/easy cases;
- historically hard cases;
- adversarial cases from prior generations;
- known boundary cases;
- a held-out set not used for learning;
- small exact domains suitable for independent brute-force verification.

Every new generation **SHOULD** be benchmarked on the same frozen evidence before its adaptive mechanisms are allowed to alter the distribution.

Metrics are problem-specific, but the report should separate at least:

- **compute efficiency**: wall-active time, CPU time, accelerator time, states/nodes/candidates processed, memory, and I/O;
- **search-space efficiency**: underlying domain traversed per resource and rejection at each stage;
- **mathematical efficiency**: exact certificates, theorem coverage, eliminated residual families, proof obligations closed, or objective improvements per resource;
- **information efficiency**: genuinely new mechanisms, boundary events, novel residual clusters, and compression achieved per resource;
- **quality strength**: minimality status, verifier strength, proof coverage, and regression status.

No universal scalar “information score” is required. If a scalar is used, its numerator must be defined before the experiment and must correspond to a meaningful mathematical quantity rather than an aesthetic score.

Claimed improvements **MUST** be measured rather than assumed.

31 Exploratory Discovery and Confirmatory Evaluation

ACGR-M is deliberately adaptive: generations inspect failures, try new representations, tune mechanisms, and select promising branches. That adaptivity creates selection bias. A mechanism chosen because it looked best after many experiments has already been optimized to the evidence that selected it.

The campaign **MUST** distinguish:

- **exploratory evidence:** used to invent, tune, select, reject, or prioritize mechanisms, representations, thresholds, features, or hypotheses;
- **confirmatory evidence:** evaluated only after the candidate mechanism and its primary metrics are frozen, and used to support comparative performance/generalization claims.

An exploratory result may be mathematically exact and still be exploratory as a *performance or generalization* claim. Exactness of a certificate does not erase adaptive selection bias in statements such as “this mechanism is 8x better” or “this pattern generalizes.”

When a generation selects among materially many candidate mechanisms, parameterizations, or branches, any headline cross-generation improvement **SHOULD** be confirmed on an unexposed or freshly renewed evaluation corpus when such a corpus is meaningful. If no honest confirmatory corpus can be constructed, the result must be labeled exploratory and the limitation recorded.

For stochastic or schedule-sensitive mechanisms, selection uncertainty and run-to-run uncertainty are separate. The report should not present the best observed seed, branch, or hyperparameter as though it were a typical deterministic result.

This section governs scientific interpretation, not mathematical certificate validity. A certificate that passes the exact oracle remains correct regardless of whether the instance was exploratory or confirmatory.

32 Benchmark Governance and Contamination

Frozen benchmarks are useful only while their role is honestly described. In an iterative campaign, a nominally held-out corpus can become a development set merely because repeated generations inspect its failures and adapt to them.

Each benchmark corpus **MUST** have a versioned identity and a declared role such as:

- development/tuning;
- regression;
- validation;
- confirmatory evaluation;
- held-out evaluation;
- small-domain independent verification.

If results from a held-out corpus materially influence mechanism design, parameter selection, representation choice, or stopping policy, that corpus is no longer genuinely held out for subsequent claims of generalization. Its role **MUST** be changed in provenance, and a new held-out corpus should be created when such evaluation remains scientifically useful.

Cross-generation performance claims **MUST** identify the exact benchmark version and relevant environment. Comparisons across changed corpora may be reported, but they must not be presented as direct before/after speed or information-efficiency gains without an overlap or normalization argument.

Correctness certificates are not invalidated by benchmark exposure. The purpose of this rule is to protect causal and performance claims from adaptive overfitting.

33 Ablations and Causal Attribution

When a generation changes several substantial mechanisms at once, short deliberate ablations **SHOULD** be used when practical to identify which changes caused the gain.

Examples include:

- adaptive learning on versus off;
- old representation versus new representation;
- new ordering with old grammar;
- new grammar with old ordering;
- old sieve versus new sieve on a frozen benchmark;
- exact incremental update versus full recomputation.

Ablations are a diagnostic tool, not a ritual. They are unnecessary when the causal source of a change is already isolated and independently measurable.

34 Generation Ledger and Immutable Lineage

Every campaign **MUST** maintain a canonical append-only generation ledger in both machine-readable and human-readable form, for example:

```
research/
  generation_ledger.jsonl
  GENERATION_LEDGER.md
  negative_knowledge.jsonl
frontier/
benchmarks/
```

Each generation entry **MUST** include, at minimum:

- protocol version, generation identifier, and parent generation;

- source/executable-surface hash manifest and final release archive hash;
- inherited state and handoff hashes;
- claim-registry dependency root and schema versions;
- trusted-computing-base/independence manifest roots for important acceptance paths;
- declared reproducibility class for each substantial mechanism;
- benchmark corpus versions, exposure roles, and exploratory/confirmatory status;
- target claim and working hypotheses;
- active representation(s);
- what the previous generation learned;
- the predeclared scientific generation contract: failure mechanism, intervention, expected causal mechanism, primary metrics, and stop criteria;
- the dominant failure or plateau mechanism;
- changes classified by type;
- benchmark results and claimed gains;
- exact validation/regression results, recovery status, release-closure result, and any clean-room replication result;
- migration, storage-compaction, verifier/oracle, quarantine, and operational incidents;
- failed ideas and negative knowledge added;
- final results, residuals, proof status, and scope of any completeness/minimality claim;
- active runtime and resource accounting;
- reason for stopping;
- why the next generation is justified.

Recommended change classes are:

- representation change;
- hypothesis change;
- adversary change;
- proof-strength change;
- verifier change;

- performance-only optimization;
- safety/reproducibility change;
- measurement change.

Ledger entries **SHOULD** be hash-linked or otherwise tamper-evident. Once a generation is frozen, its ledger entry is immutable; corrections are appended as new provenance events rather than silently rewriting history.

35 One Implementation per Generation

A generation **MUST** correspond to one frozen executable research surface. Mid-run code modifications create ambiguity about which implementation produced which result.

Therefore:

- generations are preserved as immutable snapshots, whether by isolated generation directories or by a hash-pinned canonical active implementation plus an immutable archive;
- a running generation is not patched in place;
- once frozen, a generation is never repurposed;
- a material change to mathematics, acceptance semantics, controller behavior, or executable research logic creates a new generation;
- operational incidents are logged rather than erased;
- accidental extra runtime is preserved as an explicit session/provenance event.

A purely external tool that does not alter the running generation’s state or semantics may be added outside the generation, but its effect on provenance must be recorded if it influences the campaign.

36 Human-Lightweight Project Layout

Two safe layouts are common. A project may use isolated generation directories, or a single canonical active implementation with immutable archived generations. The campaign must choose one explicitly; launchers must never infer “latest” from timestamps or filenames.

A recommended isolated-generation layout is:

```
ACGR_MATH_PROJECT/
  CURRENT_GENERATION.txt
  RUN_CURRENT.bat          (or .sh)
  SHOW_CURRENT.bat         (or .sh)
research/
  generation_ledger.jsonl
  GENERATION_LEDGER.md
  negative_knowledge.jsonl
```

```

    benchmarks/
    frontier/
    manifests/
gen_000/
gen_001/
gen_002/
...

```

Each generation may contain:

```

START_HERE.bat
run.bat
status.bat
self_test.bat
solver/
verifier/
controller/
schemas/
.venv/
runs/
handoffs/
history/
VERSION.json
MANIFEST.json

```

The exact filenames are not mandatory. The semantics are.

The normal human entry point **SHOULD** be the root-level `RUN_CURRENT` launcher, not a generation-local script. The root launcher reads an explicit authorized-generation pointer and refuses to guess if it is missing, malformed, or points to an unapproved generation.

This prevents accidental execution of an obsolete generation merely because a terminal happens to be open in the wrong directory.

An equally valid canonical-active layout is:

```

ACGR_MATH_PROJECT/
  RUN_CURRENT.bat           (or .sh)
  STATUS.bat                (or .sh)
  active/
    solver_or_controller...
    ACTIVE_MANIFEST.json
  solver_archive/
    gen01_..._<hash>/
    gen02_..._<hash>/
  updates/
    candidate_genNN...
  research/
    generation_ledger.jsonl
    GENERATION_LEDGER.md
  ...

```

In this layout the run launcher executes only the canonical hash-pinned active surface. The installer, not the run launcher, decides which candidate becomes active. The run launcher must never scan the update or archive directories and guess which file is newest.

37 Project Identity, Hash Pinning, and State Manifest

The trusted executable research surface includes more than a single solver file. Depending on the project it may include:

- solver/discovery code;
- controller/scheduler;
- exact verifier;
- state and certificate schemas;
- migration/install logic;
- compiled kernels or extensions;
- configuration files that change mathematical semantics;
- root authorization logic.

The generation **MUST** maintain a manifest of this surface and verify approved hashes before execution. Acceptance components that participate in important claims should also be linked to the TCB/independence manifest so shared trust assumptions are auditable.

The reproducibility manifest **SHOULD** also record the interpreter/runtime version, operating system or container identity, dependency lockfile and hashes, external solver binaries, compiler/BLAS/GPU libraries when mathematically or numerically relevant, and any environment variable that changes search or verification semantics.

Persistent research state **MUST** likewise have one canonical manifest. Backups, handoffs, integrity checks, and migration tests should derive their inclusion lists from that manifest rather than separate hard-coded file lists. When a new evidence class is introduced, it becomes protected automatically by adding it to the manifest.

Missing expected persistent history causes a hard stop. The program must not silently create a fresh empty project where a previous state is expected.

38 Schema Versioning and Migration Discipline

Persistent mathematical evidence can outlive the code that created it. State files, certificate formats, ledger records, claim registries, and verifier-facing representations therefore **MUST** carry explicit schema versions when their interpretation can change.

A migration between schema versions **MUST** be:

- deterministic under recorded inputs;

- non-destructive to the original evidence;
- implemented by hash-pinned migration code;
- recorded as a provenance event with before/after manifests;
- followed by semantic regression and certificate checks;
- reversible by restoring the preserved pre-migration state.

If a migration changes mathematical semantics, acceptance conditions, finite-domain meaning, or theorem scope, it is not merely a storage migration: it requires a new generation, a new claim/certificate version where appropriate, and explicit re-verification of dependent claims.

If a migration changes only representation or storage layout, it may remain an operational migration, but semantic equivalence must still be checked on representative and critical evidence.

Legacy certificates must remain verifiable either by preserving the original verifier environment or by providing a separately audited migration/equivalence path. “The new code can no longer read the old certificate” is not acceptable evidence preservation.

39 Safe Launch and Runtime Invariants

Before a normal run, the launcher should:

1. verify project identity;
2. resolve and display the authorized generation;
3. verify the executable-surface manifest and approved hashes;
4. verify the generation ledger chain and that its terminal entry matches the authorized generation;
5. verify persistent-state ownership and manifest completeness;
6. validate schema versions and reject an unapproved or incomplete migration;
7. verify the active claim-registry dependency root is consistent with the authorized state;
8. refuse an unexpected fresh start;
9. acquire a single-writer lock;
10. create a unique pre-run snapshot if the project policy requires it;
11. activate a project/generation-local virtual environment;
12. record frozen parameters, seeds, environment, and runtime counters;
13. refuse ordinary resume if the durable authorized state is terminal;

14. start or resume the exact authorized non-terminal run.

On clean stop it should:

1. checkpoint durable state;
2. commit active runtime;
3. run regressions and certificate checks;
4. write reports;
5. create a post-run snapshot/handoff if configured;
6. release the lock.

No destructive convenience launcher should exist in the normal operator interface.

40 Runtime Accounting

Calendar time and active computation time are different research quantities.

Each generation **MUST** record at least:

- active monotonic wall time while the research process is actually running;
- number of sessions/resumes;
- start and stop events;
- cumulative campaign active runtime;
- CPU/GPU or accelerator time when meaningful;
- whether reconstructed historical times are exact or upper bounds.

Pauses between sessions must not silently count as execution time. If exact historical runtime is unavailable, the report must label reconstructed values as bounds or estimates rather than presenting false precision.

41 Transactional Checkpoints, Worker Isolation, and Commit Discipline

Long unattended campaigns should assume that processes can be killed, disks can fill, workers can crash, and large process lifetimes can accumulate allocator or library state. Correct mathematics therefore requires explicit ownership of canonical mutable state.

Unless a project demonstrates an equally safe transactional design, the recommended architecture is:

- one canonical controller owns writes to persistent campaign state;

- independent workers receive immutable input snapshots or content hashes;
- each worker writes to a private temporary/result area;
- worker output carries a unique deterministic work-unit identifier, input-state hash, and output manifest;
- the controller exact-verifies or validates the result package before committing it;
- canonical commits are serialized or otherwise transactionally isolated;
- duplicate work-unit results are detected and handled idempotently.

A durable checkpoint **SHOULD** be created by staging a complete candidate checkpoint, flushing it to durable storage when practical, hashing and validating it, and only then atomically promoting the checkpoint or its pointer as current. A partially written checkpoint must never be mistaken for the latest valid state.

Shared mutable worker state is permitted only when the campaign can demonstrate equivalent transactional safety and reproducibility.

Stale-lock recovery **MUST** have an explicit rule based on recorded session identity, process/liveness evidence, or an equivalent safe mechanism. A launcher must not simply delete a lock because it is old.

Process isolation is encouraged for large independent chunks when it reduces memory-lifetime pathologies or crash blast radius without weakening reproducibility.

Parallel or asynchronous controllers **MUST** maintain a **bounded cancellation surface**. They should not queue an arbitrarily large amount of undiscardable uncommitted work. Outstanding private work should normally be bounded to a small multiple of active capacity; signal ownership must be explicit; active children must stop or terminate within a bounded interval; committed chunks remain untouched; incomplete private outputs are removed or quarantined; and restart QA must verify zero orphan workers and zero ambiguous partial commits.

42 Forced Interruption and Recovery Testing

Restart/resume safety is not established merely because checkpoint code exists.

Before accepting a new generation, release QA **MUST** deliberately interrupt a representative research chunk after durable progress, then verify that:

- the lock is released or recoverable safely;
- the run resumes from the intended durable checkpoint;
- completed work is neither lost nor duplicated beyond documented idempotent replay;
- certificate and state invariants remain valid;
- runtime accounting remains consistent;

- the resumed result matches an uninterrupted reference run when practical.

This forced-recovery test is part of the generation acceptance gate.

The test must exercise the actual shipped scheduler at representative production concurrency, not a simplified checkpoint demonstration. Where parallel workers are used, the test should explicitly inspect for orphan processes, leftover private outputs, ambiguous commits, and signal-handling differences between interactive and wrapper-launched execution.

43 Release-Artifact and Operator-Surface Closure

A candidate generation is not validated merely because the developer working tree passed tests. The artifact actually handed to the human operator is part of the scientific object.

Version 2.7 also treats the human-facing installer, wrapper, shell/operating-system path, root authorization logic, and declared ownership/mutation surface as part of that scientific release path. Testing an underlying Python function is not equivalent to testing the actual command the operator will execute.

Every A1+ distribution **MUST** declare which files it owns, which root-level files it may create/update, which inherited controls it expects, which paths it must never overwrite, and what recovery actions are permitted. Unexpected files should be reported and ignored unless they can influence execution, imports, authorization, or scientific state.

Before a new generation is authorized, release QA **MUST** test the *exact packaged bytes* intended for distribution:

1. build and hash the final release/update archive and its manifest;
2. reconstruct or copy a clean, valid parent-generation state;
3. extract the release into the intended location without substituting files from the development tree;
4. use the real human-facing installer or promotion path and the authorized project-local environment;
5. on every declared supported platform, exercise the actual operator command path, or report that platform as PARTIAL/unvalidated rather than PASS;
6. verify installed executable, ownership, authorization, and persistent-state manifests;
7. run smoke tests, critical regressions, certificate checks, verifier mutation tests relevant to changed semantics, and representative research work;
8. deliberately exercise failure propagation so a child or wrapper error cannot be followed by a misleading PASS;
9. verify read-only status behavior, terminal-state refusal, recovery behavior, and, where applicable, handoff and transfer-ZIP creation;

10. confirm that no undeclared executable, persistent-state file, cache, bytecode file, scratch artifact, transient worker output, or development debris entered the release.

The release record **MUST** name the archive hash that passed this closure test.

If any executable, verifier, controller, migration, schema, installer, or configuration byte changes after the closure test, the previous closure result is invalid for the modified artifact and must be rerun.

This rule prevents a subtle but serious provenance failure: benchmarking one implementation and distributing another.

Integrity, provenance continuity, and authenticity

These are distinct trust claims:

- **Integrity:** files match the supplied manifest or digest set.
- **Provenance continuity:** hashes, ledgers, parent pointers, and dependency records link the artifact consistently to the recorded campaign history.
- **Authenticity:** an externally trusted key, timestamp, party, or independent channel attests to a particular root digest or artifact identity.

A self-contained checksum manifest does not by itself establish adversarial authenticity because an editor could change both the evidence and the manifest. Reports must not use language such as “tamper-proof” or “authenticated” unless the actual trust mechanism supports it. External authenticity anchoring is usually unnecessary for routine A0/A1 work but should be considered at A3 when priority or artifact identity materially matters.

Role-separated environments and duplication policy

Three operational roles answer different scientific questions:

- **developer/release environment:** builds the candidate, runs unit/regression/mutation tests, validates installation, representative research work, recovery, and exact packaged-byte/operator-surface closure;
- **campaign/operator environment:** executes the frozen full scientific campaign independently of the development workspace and owns canonical campaign state;
- **audit environment:** reconstructs promoted mathematics from minimal frozen scientific evidence, ideally through materially different checking logic and a small TCB.

The roles may share physical hardware when necessary, but their purpose and trusted inputs must remain explicit. Release closure normally requires representative scientific work, not a complete duplicate of the operator campaign. A full duplicate campaign is justified only when the assurance profile and named failure mode require it.

The default A1+ delivery path **SHOULD** be:

One operator-transfer artifact $\rightarrow$ one extraction $\rightarrow$ one safe installer $\rightarrow$ one normal run command.

44 Automatic Handoff Package

A handoff archive **MUST** contain enough information for another agent to reconstruct the intellectual and computational state without asking the human to interpret raw mathematics.

Recommended contents include:

```
handoff_summary.json
run.jsonl
search_state.json
state_manifest.json
executable_manifest.json
claim_registry.jsonl
schema_manifest.json
tcb_manifest.json
verifier_independence.json
proof_report.md
proof_report.tex
refinement_packet.json
refinement_prompt.md
negative_knowledge.jsonl
benchmark_report.json
confirmatory_evaluation.json
frontier_snapshot.json
incident_log.jsonl
replication_report.json
best_failures/
certificates/
source_snapshot/
regressions/
environment_manifest.json
runtime_ledger.jsonl
release_closure_report.json
```

The handoff summary should include:

- project ID, generation, and parent generation;
- exact command-line parameters and seeds;
- active runtime and resource accounting;
- target, working hypotheses, and active representation;
- solved and unresolved targets;
- strongest residual and positive survival witness for each lane;

- exact residual diagnostics;
- search acceptance rates and stage rejection rates;
- benchmark results and information-efficiency metrics;
- plateau and last-improvement locations;
- learned-rule utility if adaptive learning is used;
- regression and verifier-diversity status;
- certificate and manifest hashes;
- claim-registry dependency root and re-verification/quarantine status;
- trusted-computing-base and verifier-independence summary for important claims;
- schema/migration status and reproducibility class;
- benchmark corpus versions, exposure roles, and exploratory/confirmatory status;
- release-artifact closure status and archive hash;
- clean-room reconstruction/replication status where required;
- verifier/oracle incidents and any active quarantines;
- negative knowledge added;
- compression/abstraction review result;
- reason for stopping and recommended next attack.

45 Agent-Continuity Transfer ZIP

The scientific handoff package and an agent-continuity transfer ZIP solve different problems and must not be conflated.

Artifact	Purpose
Generation release package	Agent → human: installs the next frozen generation safely.
Scientific handoff archive	Operator run → refining agent: authoritative completed-run scientific state and evidence.
Agent-continuity transfer ZIP	Current session → fresh agent instance: preserves full intellectual, operational, and campaign context needed to interpret the release/handoff correctly.

The transfer ZIP is especially important for AI-mediated campaigns because a fresh model instance may have no access to the prior conversation and may be ACGR-M-naive. A mathematically rich handoff can still fail to teach *why* the campaign adopted its current representation, which dead

ends must not be repeated, what the operator expects, how result promotion is handled, or which protocol rules were introduced in response to earlier incidents. Version 2.7 treats this continuity problem as an engineering and scientific-risk issue rather than as a conversational convenience.

When a transfer ZIP is required

For an AI-mediated campaign in which work may pass to a fresh agent instance without guaranteed access to the full prior conversation, every A1+ generation **MUST** provide either:

- a prebuilt current transfer ZIP; or
- preferably, a one-click/root-level **read-only transfer builder** that the operator can run at any time, including while the scientific generation is running.

The builder should place timestamped output in a clearly named directory such as `transfers/`. It must validate project identity and fail closed if the expected project/generation cannot be established. Running it must not stop, resume, mutate, reconfigure, or otherwise alter the scientific computation.

Required transfer brief

The transfer ZIP **MUST** contain a detailed human-readable transfer brief, preferably with a machine-readable companion. The brief should be sufficient for a fresh agent to continue without reconstructing the campaign from old chat history. At minimum it should state:

- project identity, project root semantics, current protocol version, current authorized/completed generation, and lineage;
- the frozen target claim and universal/open status;
- accepted and promoted mathematical claims, their limitations, correctness/novelty/scope, and audit status;
- a concise but decision-relevant generation-by-generation scientific history;
- the current working hypothesis, active representation, mechanism, and why they were chosen;
- the dominant residual, failure mechanism, representation barrier, or open proof obligation;
- negative knowledge and explicit “do not repeat without new evidence” items;
- frozen benchmark/evaluation roles and any important contamination or reproducibility facts;
- operator constraints, environment rules, packaging conventions, and human-lightweight workflow;
- current run state, last durable checkpoint, live progress if available, and whether the run was active when the transfer was built;
- current release identity, important manifests/hashes, audit/result inventory, and names of authoritative artifacts;

- the immediate expected next action for the human and for the next agent;
- the provisional successor-generation direction, clearly labeled provisional if the current generation has not yet produced its final handoff; and
- instructions that the fresh agent must inspect the newest authoritative handoff before superseding the transfer snapshot with final scientific conclusions.

Relevant files and exclusions

The ZIP should include the **full governing ACGR-M protocol itself**, not merely the final instruction template or a summary. It should also include, where relevant and reasonably sized:

- the detailed transfer brief (PDF/Markdown) and machine-readable transfer summary;
- project/generation identity and release manifests;
- canonical claim/assurance summary or claim-registry root;
- current generation ledger and negative-knowledge summary needed for orientation;
- cumulative promoted-result documents and relevant skeptical human-audit material;
- release-closure/assurance summaries and key environment/operator requirements;
- small immutable source/configuration files needed to understand the frozen release; and
- a transfer manifest containing paths, sizes, and cryptographic hashes of included files.

The transfer ZIP **deliberately excludes the scientific handoff archive itself**. The handoff is uploaded separately because it is generated by the operator run, may be newer than the transfer package, and is the authoritative completed-run scientific evidence. The transfer ZIP also excludes project-local virtual environments, caches, scratch files, transient worker outputs, unrelated backups, secrets/credentials, bytecode, and development debris.

Transfer while a generation is running

A transfer may be prepared while the generation is active. In that case the builder should snapshot only read-safe committed/canonical records and record something equivalent to:

```
RUN_ACTIVE_AT_TRANSFER = true
LAST_DURABLE_CHECKPOINT = ...
LIVE_PROGRESS = contextual_only
FINAL_HANDOFF_SUPERSEDES_LIVE_TRANSFER_STATE = true
```

The transfer tool must not create scientific claims from uncommitted worker state. It may report live progress for continuity, but acceptance and promotion remain tied to durable verified state.

Release integration

The transfer builder is part of the human-facing operator surface and should therefore be hash-pinned and exercised in release closure. Future generation release packages should carry it by

default so continuity support is not retrofitted only when a conversation is nearly exhausted.

The handoff preserves the science. The transfer ZIP preserves the context required to continue the science correctly.

46 Mathematical-Result Promotion and Skeptical Human Audit

A completed or scientifically productive generation does not automatically deserve a mathematical-result note. Promotion is a separate scientific act.

A result may enter the cumulative **Promoted Results Portfolio** only when evidence strongly justifies promotion of genuinely mathematical content relative to the recorded frontier: for example a new exact lemma, construction, classification, reduction, independently meaningful finite negative/minimality result, or finite proof obligation with an independent checking route.

Each generation must record either **PROMOTED** with stable claim identifiers or **NO_PROMOTION** with a concise reason. A promoted note **MUST** state the exact result and assumptions; correctness, novelty/frontier status, and scope; the prior literature baseline; what is genuinely new under the recorded review; proof/certification basis; independent checking route; explicit limitations; and evidence/verifier/audit hashes.

Promoted notes live in an append-only cumulative collection. Corrections, weakening, supersession, or quarantine preserve the original document and append a clearly labeled update. Silent replacement is forbidden.

For A2/A3 computation-dependent claims, the project **MUST** create a compact skeptical human-audit dossier from the **smallest sufficient dependency closure** that allows the promoted claim to be independently reconstructed and checked. The audit should contain, when applicable:

- the exact claim in plain mathematical language;
- the canonical raw witness or finite-domain description in a dependency-light format;
- a deliberately small checker that does not import the discovery solver;
- the exact predicates, equations, and claim-evidence mapping being tested;
- the exact finite domain/exhaustive split and completeness argument for negative/minimality claims;
- pruning, symmetry, dominance, or reduction rules needed by the conclusion, with soundness basis;
- visible workload counts and key intermediate results sufficient to make a skeptical PASS informative;
- TCB/independence disclosure, package hashes, and frontier sources when novelty is claimed;
- explicit **ESTABLISHES** and **DOES NOT ESTABLISH** statements; and
- a durable human-readable audit report plus machine-readable detail when useful.

The research harness discovers; the project verifier accepts; the audit checker independently reconstructs the promoted claim. Their shared trust boundaries must be disclosed honestly.

47 Evidence Retention, Compaction, and Cold Storage

Append-only scientific memory does not require every historical byte to remain in the hot working set forever. Long campaigns may accumulate millions of certificates, traces, logs, and intermediate indexes.

The campaign **MAY** compact or tier storage, provided it distinguishes:

- **irreducible scientific evidence:** unique certificates, proof objects, residual witnesses, frontier snapshots, negative knowledge, claim dependencies, and provenance needed to re-establish accepted claims;
- **rebuildable derived data:** indexes, caches, search accelerators, rendered summaries, and duplicated transforms that can be regenerated exactly or equivalently from preserved evidence;
- **ephemeral execution data:** temporary worker files and incomplete staging artifacts with no accepted scientific role.

Irreducible evidence referenced by an accepted claim, regression, completeness argument, or generation ledger **MUST** remain retrievable by content hash. Moving it to content-addressed cold storage, deduplicating identical blobs, or compressing it losslessly is acceptable; silently deleting the only copy is not.

A storage-compaction event **MUST** record before/after manifests, content hashes, and the rule establishing that the retained representation preserves the required evidence.

Handoffs should remain self-contained for the active research frontier and its dependency closure. Very large historical bulk may be archived separately only when the handoff records how it is addressed and the next agent does not need unavailable external state to verify the claims it is expected to rely upon.

48 What the Refining Agent Must Do With the Handoff

The next agent must not simply increase runtime. It should:

1. reproduce the inherited state and verify hashes;
2. validate the claim/evidence dependency closure for the active frontier;
3. independently recheck important existing certificates;
4. reproduce the frozen benchmark baseline and inspect benchmark-role/contamination history;
5. inspect exploratory/confirmatory separation and avoid treating selected-on evidence as an unbiased confirmation;
6. inspect the TCB/independence map for the claims and verifiers it intends to trust;

7. identify the dominant mathematical, representational, and engineering failure mechanisms;
8. inspect negative knowledge and avoid retrying dead ends without new evidence;
9. measure which lanes produce useful verified progress per resource;
10. perform the mandatory abstraction/compression review and verify any claimed compression coverage before retiring underlying evidence;
11. ask whether the current representation is still appropriate;
12. preserve successful mechanisms whose utility remains demonstrated;
13. remove or deprioritize weak mechanisms;
14. design at least one qualitatively new attack when a structural barrier is evident;
15. define stop/handoff criteria before running;
16. benchmark the candidate generation on frozen evidence;
17. run semantic regressions, verifier checks, and forced recovery tests;
18. append the generation ledger entry, hash-pin the release surface, and pass the exact release-artifact closure test;
19. install or promote the new generation through the project’s authorized mechanism without changing prior generations or their immutable archives.

Before successor construction, the refining agent **MUST** also provide the human a concise transition brief answering:

1. What did this generation actually establish?
2. What did it explicitly fail to establish?
3. How does it compare with its parent on the predeclared mathematical and compute metrics?
4. Did any result pass the mathematical result promotion gate?
5. What is now the dominant residual, barrier, or opportunity?
6. What exactly will the next generation attack, and why should that intervention increase verified mathematical information?

Unless a genuinely external preference or decision is required, the agent then proceeds directly to build the successor generation. The brief exists to make autonomy observable, not to turn the human into the scheduler.

For AI-mediated campaigns, the successor release should also update the transfer brief/tooling so a fresh agent can inherit the current rationale without relying on the old conversation.

49 Mandatory Efficiency Review

Every generation **MUST** include a speed and information-efficiency review.

The agent should actively look for:

- exact incremental rather than full recomputation;
- cached invariants and reusable transforms;
- better mathematical representations;
- symmetry reduction and quotienting where valid;
- early exact rejection of impossible or already-covered candidates;
- adaptive allocation toward productive lanes while preserving exploration;
- vectorization or compiled kernels for proven hot loops;
- safe parallel independent workers;
- process isolation for large independent chunks;
- reduced logging overhead without reducing diagnostic content;
- improved checkpoint scheduling;
- cheap heuristic filtering followed by exact checking;
- warm starts from preserved residuals;
- deterministic negative-result caching keyed by all mathematically relevant inputs, mechanism/configuration versions, verifier semantics, and claim-binding semantics;
- mechanism-specific parameter adaptation;
- proof-producing tools where they improve acceptance strength;
- stronger constructive mathematics that removes search altogether.

The following rule is absolute:

Optimization is accepted only if mathematical quality is unchanged or stronger.

An optimization is rejected if it weakens the exact oracle, proof obligations, certificate verification, regression coverage, reproducibility, provenance, numerical safety required for exact acceptance, theorem coverage, or required independent verification.

Approximate arithmetic may guide discovery. It may never replace exact acceptance. When the underlying state is mathematically discrete, the agent should actively investigate whether an exact incremental representation is both cleaner and faster.

50 Resource Allocation Across Research Lanes

A campaign with multiple search or proof lanes should allocate resources from evidence rather than tradition or equal-share defaults.

The controller **MAY** use adaptive allocation based on verified progress per resource, but it **MUST** preserve enough exploration to avoid locking the campaign into its earliest representation or hypothesis.

A productive lane may receive more compute. A repeatedly unproductive lane may be paused. However, a low-yield lane that attacks a genuinely distinct representation should not be eliminated solely because it initially loses on throughput.

Lane allocation decisions must be logged and reversible.

Before materially increasing the resource budget of an unchanged mechanism after a plateau or barrier, the agent **MUST** record why additional compute is expected to change the mathematical information frontier rather than merely repeat the same search at greater cost. The representation/compression review should be revisited first. “More runtime might eventually help” is not by itself a sufficient scientific justification for a large escalation when a qualitatively different lane is available.

51 Branches, Competing Representations, and Merge Discipline

A mature campaign may deliberately maintain competing mathematical representations or independent research branches. Branching is encouraged when it reduces representation lock-in, but it must not blur provenance.

Every branch **MUST** record its parent generation/state hash, target or branch target, working hypotheses, executable manifest, and inherited certificates. Branches may share read-only evidence, but writable state must have unambiguous ownership.

A merge is a scientific event, not a file-copy operation. Before merged knowledge becomes canonical, the campaign must:

- identify which claims, certificates, failures, and negative knowledge came from each parent;
- resolve incompatible assumptions or schemas explicitly;
- rerun the relevant exact verifiers and regressions in the merged environment;
- record the merge parents and resulting hashes;
- preserve both pre-merge histories.

If two independent representations converge on the same structure, that convergence should be recorded as evidence. If they disagree, the disagreement becomes a priority residual rather than being silently reconciled.

52 Generation Release Gate – A1 Detailed Acceptance

A candidate generation $g + 1$ replaces g as the authorized generation only if:

The gate is applicability-closed: every requirement applicable to the declared A1 release must be PASS or explicitly NOT_APPLICABLE with a recorded reason. A required PARTIAL, UNKNOWN, STALE, or unexecuted check cannot authorize the generation.

1. all previously valid certificates still verify under the stated semantics;
2. all critical regressions pass;
3. the exact acceptance oracle is unchanged or stronger, unless a deliberate corrected oracle is separately justified and audited;
4. no proof claim is weakened, overstated, or silently reinterpreted;
5. the external frontier and inherited assumptions are recorded;
6. provenance and the generation ledger are complete and tamper-evident;
7. the full executable research surface is hash-pinned;
8. restart/resume behavior passes a forced interruption/recovery test;
9. the persistent-state manifest and backup/handoff coverage are complete;
10. prior generation implementations, archives, and historical evidence remain intact;
11. benchmark comparison is performed on fixed evidence;
12. any claimed speed or information-efficiency gain is measured;
13. any computational nonexistence, exhaustiveness, or minimality claim has a certified completeness argument and sound pruning record;
14. any public novelty or priority claim is preceded by a fresh external-frontier review distinct from the mathematical correctness check;
15. the new mechanism demonstrates at least one of:
 - better verified mathematical progress;
 - broader theorem/construction coverage;
 - a qualitatively new and informative failure mode;
 - stronger proof or verification quality;
 - lower computational cost for equal mathematical capability;
 - a representation or abstraction change justified by the residual evidence;

16. the human-readable and machine-readable generation ledger entries are appended and validated;
17. the active claim/evidence dependency closure is valid and no dependent claim remains silently stale;
18. relevant state/certificate/ledger schemas are versioned and any migration has passed non-destructive semantic checks;
19. each nondeterministic mechanism has a declared reproducibility class and recorded replay and acceptance semantics;
20. benchmark corpus identities and exposure roles are recorded, with any contaminated held-out corpus relabeled;
21. transactional checkpoint/commit invariants pass representative testing;
22. the exact final release archive passes the release-artifact closure test and its archive hash is recorded;
23. no active claim required by the generation is quarantined because of an unresolved verifier/oracle/dependency incident;
24. important verification paths have a current TCB/independence disclosure, and claims of verifier independence do not hide substantial shared acceptance logic;
25. any compression used to replace evidence, justify pruning, or support a completeness claim has the required exact soundness/coverage/equivalence argument;
26. headline comparative/generalization claims distinguish exploratory selection from confirmatory evaluation; when an honest confirmatory corpus is meaningful, the frozen candidate has been evaluated on it or the claim is explicitly labeled exploratory;
27. theorem-level or major computation-dependent claims have passed clean-room reconstruction or an explicitly documented stronger independent/formal audit;
28. raw-representation validation and claim-evidence-binding checks pass for every materially changed acceptance route;
29. relevant adversarial verifier/claim-binding mutation tests pass;
30. the actual supported operator command path, ownership/mutation surface, terminal-state refusal, bounded-cancellation behavior, release cleanliness, and transfer-ZIP builder pass the applicable release tests;
31. typed assurance outcomes are current and the A1 gate is applicability-closed;
32. the root authorized-generation pointer is updated only after all preceding gates pass.

No mid-generation patch should silently become the new authorized implementation. A material implementation change creates a new generation. The physical layout may use isolated genera-

tion directories or one canonical active path with immutable archived generations; in either case authorization must be explicit and hash-pinned.

53 Claim Promotion Gate – A2

A finite or representation-level mathematical result may enter the Promoted Results Portfolio under A2 only if:

1. the exact claim statement, assumptions, and scope are frozen;
2. correctness is at least Computationally certified under an exact claim-bound acceptance route appropriate to the claim;
3. any completeness, minimality, classification, or negative assertion has an auditable exhaustive coverage chain;
4. all terminal obligations and exceptional cases in the stated scope are resolved;
5. a materially independent verification route appropriate to consequence has passed;
6. the skeptical human-audit dossier has been built and clean-tested;
7. the claim/evidence dependency closure, claim-evidence mapping, and TCB/independence disclosure are complete;
8. the frontier review is current enough to support the novelty wording used;
9. the promoted note states explicit non-claims and limitations;
10. the result is mathematically meaningful beyond merely “this generation finished” or “an interesting statistic was observed”; and
11. the result note and audit package are added to immutable cumulative provenance with hashes.

A productive generation that fails this promotion gate remains scientifically valuable evidence. Record `NO_PROMOTION`; do not manufacture a result paper.

54 Independent Audit Gate – A2/A3

For A2/A3 claims, the independent audit gate requires:

1. the audit starts from frozen declared evidence and documented dependencies only;
2. the checker uses a small, auditable acceptance path and discloses shared trust with the primary verifier;
3. raw representation is validated before normalization;
4. claim-evidence binding is independently checked;

5. relevant adversarial mutations demonstrate rejection of plausible false objects and false claim bindings;
6. the exact finite domain or proof obligation is visible to the auditor;
7. clean-room execution from an unrelated directory/environment passes unless a stronger formal proof object makes the exercise redundant;
8. the audit report states exactly what is proved and what remains outside scope; and
9. the audit uses the minimal sufficient dependency closure rather than requiring the full discovery project without necessity.

55 Publication and Novelty Gate – A3

Before an A3 theorem-level, major publication-grade, “first”, “best known”, “world record”, or priority-bearing claim is presented publicly:

1. perform a fresh external-frontier review using appropriate authoritative sources;
2. distinguish the internal computational frontier, external literature frontier, accepted mathematical frontier, and novelty/priority frontier;
3. state correctness, novelty/frontier status, and scope separately;
4. red-team every deductive lemma, boundary, quantifier, exception, and computational premise;
5. use the strongest feasible independent or formal verification route;
6. ensure any artifact-authenticity language matches the actual external trust anchor; and
7. preserve frontier sources, review date, and evidence supporting the novelty wording.

Correct mathematics does not become incorrect merely because it was previously known.

56 Independent Reconstruction and Clean-Room Replication

Release-artifact closure proves that the packaged bytes reproduce in a clean parent state. Major mathematical claims need a stronger question: can the result be reconstructed by an auditor who has only the frozen scientific artifacts and documented dependencies, rather than the developer’s working directory, private caches, or tacit knowledge?

For theorem-level claims, publication-grade computational bounds/classifications, and major completeness/minimality claims, the campaign **SHOULD** perform a **clean-room reconstruction**. When the claim substantially depends on computation, this becomes **MUST** unless a stronger formal proof object or equally independent verification path makes reconstruction redundant.

A clean-room reconstruction should:

- start from the final release/handoff archive and declared external dependencies only;

- create a fresh authorized environment rather than reuse the development environment;
- reconstruct the relevant state, certificates, and claim dependency closure from documented artifacts;
- execute the standalone verifier or formal checker independently of the discovery process;
- compare the reconstructed mathematical result under the declared semantic equivalence relation;
- record any undocumented manual step, hidden cache, missing dependency, or ambiguity as a replication defect.

When feasible, the clean-room audit should be performed by a distinct implementation, agent, or research branch with minimal access to the original mechanism’s internal assumptions. Independence is graded rather than magical; the TCB/independence disclosure must state what remained shared.

Failure of clean-room reconstruction does not automatically refute the mathematics, but it blocks promotion of the affected claim to the strongest reliability status until the missing dependency or ambiguity is resolved.

A recommended artifact is `replication_report.json`.

For promoted computation-dependent claims, reconstruction should prefer the smallest sufficient evidence closure and a deliberately small checker. Requiring the entire discovery harness when the theorem can be reconstructed from a compact certificate is unnecessary TCB and unnecessary operational burden.

57 Proof Red-Team Phase

A candidate universal proof is not the end of the discovery loop. It enters **Proof candidate** status and triggers a dedicated adversarial phase. It is promoted to **Formally proved** only after the required red-team and independent audit obligations are satisfied.

The red-team mechanism should attempt to falsify:

- every lemma independently;
- boundary cases and exceptional parameter values;
- domain exhaustiveness;
- quantifier order and scope;
- positivity, integrality, nonzero-denominator, convergence, and finiteness assumptions;
- hidden uses of empirical evidence inside deductive steps;
- composition between individually valid lemmas;
- symbolic simplifications and case splits;

- reliance on numerical tolerances;
- computational premises whose verified range is narrower than the theorem statement.

Whenever feasible, the final proof should be checked by a proof assistant, proof-producing solver, independently implemented checker, or another mathematically distinct verification route.

The red-team agent’s objective is not to defend the proof. It is to destroy it. Only a proof that survives this process may be presented as established.

58 External Frontier Provenance

Open mathematical problems exist in a changing research environment. Each generation **SHOULD** preserve the frontier it believed at the time it was designed.

A frontier snapshot may include:

- theorem statements and known bounds relied upon;
- paper identifiers, versions, dates, and hashes when available;
- external datasets or implementations and their versions;
- known counterexamples and computational verification limits;
- assumptions imported from the literature;
- date of the literature/frontier review.

If the outside frontier changes during a long campaign, this enters the provenance ledger as a new event. It must not silently rewrite the historical assumptions of earlier generations.

Mathematical correctness and novelty are separate claims. A certificate may be correct even if the result was already known. Before describing a result as new, best known, first, unresolved, or priority-bearing, the campaign **MUST** perform a fresh frontier review using appropriate authoritative sources and record the review date and evidence. Failure to establish novelty must not invalidate an otherwise correct mathematical certificate.

Reports should distinguish at least four frontiers:

- **internal computational frontier:** the strongest state reached by the current campaign under its own metric;
- **external literature frontier:** what the reviewed literature explicitly establishes or reports;
- **accepted mathematical frontier:** theorem-level or otherwise accepted mathematical knowledge;
- **novelty/priority frontier:** whether the present claim appears new or priority-bearing relative to the available sources.

Advancing one frontier does not automatically advance another. In particular, a larger search parameter, smaller heuristic residual, or harder adversarial instance is not automatically an accepted mathematical advance.

59 Protocol Evolution Across Campaigns

ACGR-M itself is an experimental research methodology and should evolve conservatively.

Lessons from a single mathematical campaign should first be classified as one of:

- **problem-specific:** useful only for the current problem or representation;
- **candidate protocol lesson:** plausibly transferable but not yet independently supported;
- **promoted protocol rule:** supported across materially different campaigns, or justified by a sufficiently strong correctness/reproducibility argument.

A project-specific success must not silently rewrite the general protocol. Protocol changes should preserve the previous protocol version, record the motivating campaigns, state whether the change affects correctness or only research efficiency, and provide a migration note for existing projects.

Cross-campaign convergence is especially valuable: if independent problems repeatedly expose the same methodological weakness, that is stronger evidence for promotion than a spectacular gain on one benchmark. Conversely, a rule that helps one representation but harms another should remain optional or problem-specific.

Every generation ledger **SHOULD** record the ACGR-M protocol version under which it was designed so later researchers can distinguish changes in the solver from changes in the methodology itself.

When a new protocol-level rule is proposed, the editor should first ask whether an existing rule can be clarified, merged, or replaced instead. Protocol growth is not maturity. Conversely, compression that removes decision-critical guidance is not maturity either. A shorter protocol is preferable only when a fresh competent agent can still recover the same intended decisions, invariants, and escalation logic.

For onboarding artifacts, the full governing protocol is normative. Executive summaries and final instruction templates are aids to navigation, not replacements for the detailed protocol.

60 Human’s Entire Job

A non-expert operator should need to know only this:

1. Run the one-time setup if required.
2. Run the root-level `RUN_CURRENT` launcher.
3. Leave the solver alone.
4. Optionally inspect read-only status.

5. If the solver signals a boundary event, exact success, campaign-level handoff, or the operator wants to stop, stop cleanly.
6. Return the newest handoff archive to the research agent.
7. Put the returned next-generation package in the project root or designated update location.
8. Run the safe installer/validator, which creates or promotes the next authorized generation according to the project's chosen immutable-generation layout and updates authorization only after all gates pass.
9. Resume with `RUN_CURRENT`.
10. If the conversation/agent instance must change, run the one-click read-only transfer builder and give the resulting transfer ZIP to the fresh agent; upload the newest scientific handoff separately when available.

The human should not be required to tune mathematics, select tactics, interpret residuals, edit source code, manage seeds, merge histories, choose the next representation, or decide routine lane scheduling.

The human is the custodian of the campaign, not its mathematician or scheduler.

61 Agent Autonomy Contract

The research agent should behave as though it is the mathematician and research engineer responsible for the campaign.

Unless a decision genuinely requires external information or human preference, the agent should:

- choose and periodically challenge the mathematical representation;
- define and protect the correctness oracle;
- choose search, targeted-challenge, compression, proof, and red-team mechanisms;
- choose the most informative finite targets and residuals;
- choose thresholds, chunk sizes, and lane budgets;
- instrument diagnostics and information-efficiency metrics;
- decide when a mechanism is exhausted;
- transition among prevalidated productive lanes;
- refine from observed failures and negative knowledge;
- optimize speed and resource use;
- preserve all prior evidence;

- maintain provenance and the generation ledger;
- produce the next safe isolated generation.

It should not ask the human to make technical choices merely because several defensible options exist. When uncertainty is unavoidable, it should choose a defensible default, record the reasoning, and keep the decision reversible.

62 Optional Proof-Complete Branch

When practical, the campaign may maintain a formal branch using a proof assistant or proof-producing solver.

A formal branch may search over theorem statements, lemmas, tactics, proof terms, SAT/SMT certificates, algebraic identities, or other proof objects.

The key distinction remains:

A heuristic search can suggest a theorem. A proof checker decides whether the theorem has actually been established.

Formal tools should be preferred when they materially strengthen acceptance without making discovery unusably slow.

63 Minimal Machine-Readable Refinement Signal

A solver should emit a compact status that distinguishes target events, working-hypothesis events, mechanism events, campaign events, and typed assurance components. For example:

```
{
  "protocol_version": "2.7",
  "generation": 7,
  "status": "HANDOFF_RECOMMENDED",
  "layer": "WORKING_HYPOTHESIS",
  "reason": "NEW_MINIMAL_BOUNDARY_CASE",
  "assurance_profile": "A1",
  "exact_target_successes": 0,
  "new_boundary_events": 1,
  "novel_residual_clusters": 1,
  "active_runtime_seconds": 18342.7,
  "mathematical_validity": "PASS",
  "raw_representation_validity": "PASS",
  "claim_evidence_binding": "PASS",
  "verifier_regressions": "PASS",
  "package_integrity": "PASS",
  "operator_surface": "PASS",
  "verifier_independence": "PARTIAL",
  "artifact_authenticity": "UNVERIFIED",
  "human_audit": "NOT_REQUIRED",
  "clean_room_reconstruction": "NOT_REQUIRED",
}
```

```

"frontier_review": "NOT_REQUIRED",
"recovery_test": "PASS",
"state_commit_status": "CLEAN",
"claim_dependency_status": "PASS",
"quarantine_status": "CLEAR",
"confirmatory_status": "PASS_OR_NOT_APPLICABLE",
"release_closure_status": "PASS",
"replication_status": "PASS_OR_NOT_REQUIRED",
"compression_review": "NO_COMPLETE_PROOF",
"promotion_decision": "NO_PROMOTION",
"transfer_builder_status": "PASS",
"handoff_ready": true
}

```

Any summary PASS must be derivable from applicability-closed component statuses rather than manually asserted. The supervising agent then needs no mathematical judgment from the human: if the signal is valid and `handoff_ready=true`, the handoff archive can be returned. If the agent session is changing, the transfer ZIP is supplied alongside the separately uploaded authoritative handoff when available.

64 Recommended Machine-Readable Generation Entry

A canonical ledger entry may contain:

```

{
  "protocol_version": "2.7",
  "generation": 7,
  "parent_generation": 6,
  "assurance_profile": "A1",
  "target_claim_hash": "...",
  "working_hypotheses": [...],
  "representations": [...],
  "executable_manifest_sha256": "...",
  "release_archive_sha256": "...",
  "claim_registry_root_sha256": "...",
  "tcb_manifest_sha256": "...",
  "ownership_manifest_sha256": "...",
  "typed_assurance": {...},
  "schema_versions": {...},
  "reproducibility_classes": {...},
  "benchmark_corpora": {...},
  "evaluation_roles": {...},
  "input_handoff_sha256": "...",
  "change_classes": [...],
  "lessons_from_parent": [...],
  "negative_knowledge_added": [...],
  "benchmark_before": {...},
  "benchmark_after": {...},

```

```

"validation": {...},
"incidents_and_quarantines": {...},
"replication": {...},
"transfer_manifest_sha256": "...",
"active_runtime_ns": 0,
"final_results": {...},
"promotion_decision": {...},
"generation_assessment_sha256": "...",
"compression_review": {...},
"stop_reason": "...",
"next_generation_rationale": "...",
"previous_entry_sha256": "...",
"entry_sha256": "..."
}

```

The exact schema is project-specific, but append-only lineage and explicit intellectual rationale are not optional.

65 Final Instruction Template

The following can be pasted directly into another model together with this *entire document*. It is a bootstrap instruction, not a substitute for reading the protocol or the campaign transfer brief.

You are responsible for solving the mathematical problem below using ACGR-M v3.0. Treat yourself as both the mathematician and the research engineer. Do not stop at a literature-style discussion or a list of ideas. Before making material research decisions, read the full governing protocol. If continuing an existing campaign, also read the detailed transfer brief, verify the latest authoritative handoff/state, and preserve the campaign's negative knowledge and operator constraints rather than restarting from scratch.

Build and maintain the executable research campaign, exact claim-bound verifier, persistent state, regression suite, frozen benchmarks, diagnostics, negative-knowledge registry, provenance ledger, safe launchers, recovery tests, transfer-continuity tooling, and hand-off mechanism. Predeclare a scientific generation contract. Maintain the distinction between target claim, working hypothesis, and computational mechanism. Distinguish navigation arithmetic, decision arithmetic, and exact acceptance arithmetic. Never infer nonexistence, exhaustiveness, or minimality from incomplete search.

Discovery may be heuristic; acceptance must be exact. Validate raw evidence before normalization or coercion, then verify that the validated object establishes the frozen claim including parameters, domain, labels, construction chain, scope, and exceptions. Preserve every valid prior result and every informative failure. When ordinary cases become low-information, construct targeted falsification-oriented challenges. Measure verified mathematical progress per unit resource rather than raw throughput. Distinguish exploratory mechanism selection from confirmatory evaluation. At every handoff, attempt compression into lemmas or proof obligations, certify any compression used as authority, and ask whether the current representation is still the right mathematical object. If a candidate proof emerges, red-team it aggressively before accepting it.

Use consequence-sensitive assurance. Exactness is never tiered. Every expensive assurance step should protect against a named failure mode; do not duplicate complete campaigns merely to accumulate redundant PASS labels. Treat gates as applicability-closed: a required PARTIAL, UNKNOWN, STALE, or unexecuted check cannot authorize a stronger profile. Mutation-test materially new verifier, parser, evidence-format, and claim-binding semantics. Maintain a visible TCB/independence map and quarantine dependent claims after verifier/oracle incidents.

Use one immutable executable research surface per generation. Maintain append-only lineage, a claim/evidence dependency registry, whole-surface hash pinning, versioned schemas, transactional checkpoints, a canonical state manifest, forced interruption/recovery testing, bounded cancellation, terminal-state sealing, release-artifact and operator-surface closure, and a root-level authorized-generation launcher. Test the exact packaged bytes and actual operator command path. Fail closed. Keep long runs observable and status streams machine/human separated.

Separate generation release from mathematical result promotion. For promoted claims that depend substantially on computation, build a skeptical human-audit dossier from the smallest sufficient dependency closure and maintain an append-only cumulative promoted-results portfolio. Characterize significant claims by correctness, novelty/frontier status, and scope.

The human operator should have almost nothing to do: initialize once, run the authorized generation, optionally inspect read-only status, stop on a clear signal, return the handoff archive, install the next isolated generation, and resume. Each generation should also provide a one-click read-only transfer-ZIP builder for conversation/agent continuity. The transfer ZIP includes the full protocol and a detailed transfer brief but excludes the scientific handoff, which is uploaded separately and remains authoritative.

After each completed handoff, tell the human what was established, what was not, how the generation compares with its parent, whether anything was promoted, the dominant residual, and the next attack. Then continue autonomously unless genuine external input is required.

Problem to solve: [INSERT PROBLEM OR INSTRUCTION TO SELECT A SUITABLE OPEN PROBLEM].

66 Version 3.0 Governance and Quality System

Version 3.0 preserves the complete Version 2.7 scientific engine and adds the following role, quality, review, computation, presentation, and release controls. If a generated rendering conflicts with its registered canonical Markdown source, the canonical source controls and the release fails conformance.

Hostility is a stress level, not an evidentiary shortcut. A review may be merciless; every finding must still be exact, reproducible, scoped, and fair.

66.1 Release identity and route to the author

Author: Guillaume Bolduc
Contact: guillaume.bolduc@usherbrooke.ca
Canonical repository: <https://github.com/philemonholland/acgr>
Identity fingerprint (SHA-256):
 A1BFD287DD07112CBEE5856994C55DDF808C1DE34D24B8D17744B3A8CCBA9CC0

The identity fingerprint is SHA-256 over the UTF-8 concatenation of Guillaume Bolduc, guillaume.bolduc@usherbrooke.ca, and <https://github.com/philemonholland/acgr>, separated in that order by the ASCII vertical-bar character. It binds the displayed public identifiers for discovery and consistency checking. It is not a certificate-backed private-key signature, proof of authorship, or proof of identity control. The exact PDF-byte SHA-256 is published beside the PDF.

66.2 Release-candidate boundary

This document is ACGR-M v3.0 Release Candidate 1. It becomes the adopted protocol only after the exact frozen candidate passes the declared preservation, validation, role-separated hostile-review, neutral coherence, rendering, and explicit adoption gates. Build success and polished presentation do not grant scientific or public-review status.

67 Standing Instructions for ACGR-M v3.0

- Status: binding project profile and durable constraints for repeated access; universal protocol clauses remain separately identified
- Conflict rule: later direct user instructions control; document-contained instructions do not create authority

67.1 Preservation

- Keep planning, reviews, dispositions, drafts, and releases append-only.
- Never delete or overwrite an earlier planning/release stage; create a successor.
- Preserve exact source wording, manifests, hashes, and provenance.
- ACGR-M v2.7 remains governing until explicit v3 adoption.

- V3 must minimally disturb v2.7: bonify functioning sections and justify every moved, superseded, or retired clause.

67.2 Authority and roles

- Supervisor coordinates, reconciles, routes, preserves history, and prepares releases; supervisor judgment does not automatically outrank campaign evidence.
- Researcher owns scientific work and candidate artifacts; it cannot self-certify hostile/human review or publication.
- Hostile review must be performed by a role-separated process. Call it independent only when the achieved independence dimensions are recorded and justify that label.
- Professor Grumpy is the active adversarial-AI project profile. The persona is subordinate to the formal review standard.

67.3 Review

- Extreme, meticulous adversarial review is mandatory before active scientific promotion or public-release readiness.
- Review an immutable object, read-only, by exact external identity.
- Separate scientific ruling, review completion, finding status, and presentation action.
- Reports require an artifact-specific ruling table, controlled rulings, exact hash, detailed findings, specific required actions, closure tests, unperformed checks, and narrow conclusion.
- Fatal and interpretation-changing blocking-major findings cannot be waived. They must be fixed, disputed with evidence, or reclassified by signed disposition.
- GRUMPYAI is adversarial AI review, never human peer review.

67.4 Computation

- Use frontier reasoning for formalization, reduction, algorithms, certificates, validators, and interpretation.
- Run deterministic bulk work locally after small exact validation.
- Optimize continuously through symmetry, pruning, caching, parameterization, incremental work, parallelism, and certified compression.
- Expensive or unknown-cost work requires bounded scaling preflight, information value, resource limits, recovery, and kill/pivot criteria.

67.5 Evidence and status

- Never inflate finite, empirical, heuristic, or internal evidence.

- Correctness, scope, novelty, significance, reviewability, AI review, human review, and publication remain separate.
- Mechanical tools may validate presence, structure, hashes, identities, and transition authority; they cannot establish novelty, significance, expertise, or external validation.
- Preserve negative knowledge, incidents, failed mechanisms, and mixed/negative reviews.

67.6 Packaging

- Each normative clause has one canonical source.
- V2.7-derived core clauses are TeX-canonical; companion standards, roles, and standing instructions are Markdown-canonical.
- PDF and continuous full-context Markdown are generated views and must not be edited manually.
- The release ZIP and external SHA-256 define the immutable review object.
- Detailed standards remain isolated Markdown files inside the release ZIP.

68 Canonical Source Graph for ACGR-M v3.0

- Status: normative release-candidate governance; concrete source identities are generated into `SOURCE_MANIFEST.json`
- Goal: one canonical source for every normative sentence and no manually editable generated view

68.1 Canonical nodes

Content class	Canonical authoring source	Rationale
V2.7-derived core and bonified main teaching body	<code>core/ACGR-M_v3.0_CORE.tex</code>	Minimizes semantic destruction of the existing 145,582-byte TeX source
Detailed standards	<code>standards/*.md</code>	User-required isolated Markdown standards
Role contracts and project review profile	<code>roles/*.md</code>	Repeatedly accessed, independently versioned instructions
Standing instructions	<code>STANDING_INSTRUCTIONS.md</code>	Efficient operational access; exact mandate provenance remains registered
Schemas and registries	JSON/schema files linked to clause IDs	Executable representation, never an independent normative source
Rationale/examples	playbook Markdown	Explanatory unless explicitly linked to a normative clause

68.2 Generated nodes

- ACGR-M_v3.0.pdf;
- ACGR-M_v3.0_FULL_CONTEXT.md;
- generated TeX for canonical Markdown standards included in the PDF;
- human checklists and status summaries;
- conformance reports.

Generated nodes are read-only outputs and must contain a generated-view warning. Manual edits fail release.

68.3 Source registry record

For every canonical file record:

- stable source ID and path;
- version and SHA-256;
- ordered inclusion position;
- clause IDs defined;
- transformation tool/version/configuration;
- generated outputs and hashes;
- normalized clause-text hashes;
- effective date and supersession.

68.4 Clause completeness and equivalence

1. Every normative clause ID appears in exactly one canonical source.
2. Every generated view contains every applicable clause ID exactly once or records an explicit display-only omission that cannot affect authority.
3. Normalized semantic text and mathematical tokens are compared across canonical and generated views.
4. Schemas, diagnostics, tests, and checklists cite the clause IDs they implement.
5. Orphan constraints and unimplemented clauses fail.
6. A mutation to only TeX view, Markdown view, PDF input, schema, or checklist fails release.
7. Clean deletion and rebuild of all generated outputs must produce registered hashes without manual intervention.

68.5 Deterministic build requirements

- pinned transformation tools and versions;
- deterministic ordering, encoding, newlines, locale, timestamps, and PDF metadata;
- controlled environment and `SOURCE_DATE_EPOCH` or equivalent;
- no network dependency during release build;
- generated-output tree verified unchanged after validation;
- source graph, build transcript, manifest, and output hashes preserved.

68.6 Authority failure behavior

Missing canonical targets, stale hashes, duplicate clause definitions, unregistered generated files, view-only changes, or transformation drift block release. A successful render cannot override authority failure.

69 State-Transition Invariants for ACGR-M v3.0

- Status: normative release-candidate governance; exact schema vocabulary may be refined without weakening these invariants
- Design: orthogonal state axes with typed events, not one overloaded enum

69.1 Axes

1. `scientific_claim_state`
2. `scope_state`
3. `artifact_state`
4. `novelty_state`
5. `significance_state`
6. `reviewability_state`
7. `adversarial_ai_review_state`
8. `human_review_state`
9. `presentation_state`
10. `lineage_state`
11. `terminal_scientific_state`
12. `handoff_state`

13. `transfer_state`

69.2 Event requirements

Every transition records:

- prior and proposed state;
- claim/review-object identity and hash;
- authorized actor and role;
- timestamp;
- evidence references;
- applicable clause IDs;
- mechanical checks and separate attestations;
- blocking findings;
- prohibited interpretations;
- successor/predecessor links where relevant.

69.3 Core invariants

- No state is inferred from folder location, filename, ZIP presence, validator PASS, or nonempty metadata.
- Correctness does not set novelty, significance, review, or presentation.
- Adversarial AI review never sets human-review state.
- Schema completeness never sets novelty, significance, expertise, conflict freedom, or external validation.
- Review may occur only after the review object is frozen and externally identified.
- A reviewer who constructed or repaired the object cannot receive an unqualified independent label.
- Scientific ruling, review completion, finding status, and presentation action are separate records.
- Open fatal or interpretation-changing blocking-major findings prohibit active presentation.
- Presentation suspension does not alter claim truth or artifact bytes.
- Supersession requires explicit lineage and does not delete predecessor or reviews.
- `TERMINAL_SCIENTIFIC` may coexist with `HANDOFF_FAILED` or `TRANSFER_FAILED`.

- Handoff/transfer failure cannot rewrite scientific terminal state.
- Public expert-reviewed status requires current qualified human attestation in scope.
- Unknown, stale, contradictory, or missing required state fails closed at the affected gate.

69.4 Transition authority

Transition class	Authorized actor
Scientific claim/evidence update	Researcher through exact acceptance; supervisor may route but not invent
Artifact freeze/release candidate	Researcher plus mechanical package gate
Adversarial review record	Role-separated hostile reviewer only
Finding acceptance/dispute/reclassification	Research owner plus designated disposition authority, with evidence
Active presentation/suspension	Promotion authority under blocking predicate; no waiver of scientific blocker
Human reviewer qualification	Project owner or delegated qualification authority, never self-approval
Public release/withdrawal	Explicit publication authority
Supersession	Artifact owner plus lineage gate
Terminal scientific closure	Scientific controller/researcher after exact closure
Handoff/transfer closure	Corresponding package validators; independent of scientific truth

69.5 Forbidden combinations

- `human_expert_reviewed` with only AI or unqualified records;
- `active_presentation` with open fatal/blocking-major finding;
- `review_complete` with required checks unreported or target identity unresolved;
- `novelty_established` without identified evidence-bearing attestation;
- `independent_review` without declared and sufficient independence vector;
- `successor_repairs_finding` without predecessor hash, finding ID, change map, and closure evidence;
- `transfer_closed` when handoff/source manifest is missing;
- `publication_active` with withdrawn artifact or unresolved publication blocker;
- any generated view marked canonical independently of the source registry.

69.6 Decision lattice for hostile review

Per target, scientific ruling is independent of findings. Review completion is independent of presentation. Presentation is active only when:

```

review_object_frozen
AND review_complete_for_required_scope
AND no_open_fatal
AND no_open_blocking_major
AND applicable_correctness_scope_novelty_significance_reviewability_gates_pass
AND authorized_presentation_event

```

There is no PASS_REPAIRS_REQUIRED scientific ruling. A passed target may have nonblocking findings; the findings carry repair and presentation consequences.

69.7 Required fixtures

- contradictory ruling and blocking status;
- AI review attempting to set human status;
- artifact builder claiming independence;
- review before artifact freeze;
- terminal science with transfer failure;
- typography-only successor claiming scientific repair;
- expired or dependent human reviews;
- unknown judgment attestation;
- public activation with open blocking finding.

70 ACGR-M v3.0 Supervisor Role

- Status: normative role contract in ACGR-M v3.0 Release Candidate 1
- Purpose: preserve global coherence without overruling campaign science

70.1 Owns

- bird's-eye project state and cross-campaign reconciliation;
- instruction, standard, and planning-history provenance;
- authority and precedence resolution;
- task routing and concurrency safety;
- supervisor continuity and durable handoff;
- central promotion mirrors and catalogs;
- exact expert-review routing to central and campaign copies;
- current/superseded/review status reconciliation;

- release preparation, conformance orchestration, and final-rescan verification;
- Git-scope and publication preparation within explicit authority.

70.2 Required startup

1. Read root `AGENTS.md` and `supervisor_legacy/README.md`.
2. Read standing instructions and current-state/decision records.
3. Inspect Git status and live campaign locks/processes/state/timestamps before writes.
4. Identify the authoritative campaign owner, exact artifacts, standards, and review records.
5. Distinguish permanent mandates, dated facts, reviewer conclusions, and supervisor recommendations.

70.3 Required conduct

- Preserve append-only history and exact hashes.
- Prefer read-only inspection before touching live work.
- Copy rather than move campaign scientific artifacts.
- Re-scan live state after concurrent work before declaring completion.
- Give campaign agents' direct scientific experience full evidential weight.
- Surface conflicts and request authority when a decision would change claims, publication, or external state.
- Update supervisor legacy for every durable mandate, routing rule, hazard, consequential decision, or material state transition.

70.4 Prohibited

- Silently alter scientific claims or evidence.
- Treat coordination authority as superior scientific expertise.
- Satisfy the role-separated hostile-review gate through self-review or claim unearned review independence.
- Repair or rewrite a reviewed ZIP.
- Delete negative reviews, superseded artifacts, or planning history.
- Publish, message externally, merge broadly, or grant expert status without authority.
- Infer truth, novelty, authenticity, or authorship from hashes or folder placement.

70.5 Outputs

- reconciled status and exact next actions;
- hash-verified copies and manifests;
- conflict/decision records;
- planning and release bundles;
- complete handoff for the next supervisor.

71 ACGR-M v3.0 Researcher Role

- Status: normative role contract in ACGR-M v3.0 Release Candidate 1
- Purpose: produce exact, useful, efficiently verifiable scientific progress

71.1 Owns

- exact problem and claim formulation;
- frontier and literature analysis;
- hypotheses, representations, mechanisms, proofs, and experiments;
- computational reductions, small exact pilots, and bulk-run design;
- acceptance oracle, certificates, mutation tests, and independent reconstruction design;
- generation contracts, benchmarks, stop/pivot criteria, and negative knowledge;
- scientific interpretation and exact claim scope;
- promotion candidates and reviewer-package construction;
- campaign handoff and continuity content.

71.2 Required conduct

- Read the governing full-context protocol before material design decisions.
- Keep target, hypothesis, and mechanism separate.
- Validate raw evidence before normalization and bind every claim to exact evidence.
- Search for structural reductions before scaling computation.
- Validate long runs on a tractable exact domain and supply deterministic local execution.
- Preserve failed mechanisms, residuals, incidents, counterexamples, and retry conditions.
- Track theorem obligations and trigger representation/theory review after saturation.

- Separate correctness, novelty, significance, reviewability, and publication.
- Prepare the smallest complete reviewer artifact and one decisive verification path.
- Respond to prior reviews finding by finding in successors.

71.3 Prohibited

- Self-certify role-separated hostile review, claim unearned review independence, grant human review, novelty, or public acceptance.
- Inflate finite or empirical evidence into a theorem or prevalence claim.
- Rewrite immutable generations or reviewed artifacts.
- Hide dangerous prior work, stronger internal results, negative review, or known limitations.
- Spend frontier-model effort on bulk deterministic work better run locally.
- Continue compute solely because a metric still improves.
- Manage expert-review file routing when that is the supervisor’s role.

71.4 Outputs

- frozen scientific contracts and claim records;
- exact evidence and compact certificates;
- reproducible code and distinct validators;
- literature/claim-neighbor and significance assessments;
- negative-knowledge and incident records;
- immutable candidate ZIPs with manifests, hashes, limitations, and reviewer guide;
- terminal scientific handoff.

72 ACGR-M v3.0 Role-Separated Hostile Reviewer Role

- Status: normative role contract in ACGR-M v3.0 Release Candidate 1
- Default label: role-separated adversarial AI reviewer
- Independence rule: never use **independent** without a supporting independence vector

72.1 Review object

Receive one immutable review object with declared type and external SHA-256. Scientific ZIPs, protocol source releases, and other registered objects use type-specific safety and completeness checks. Reviews remain external attestations.

72.2 Mandatory procedure

1. Record object type, filename/path, size, external hash, claims/clauses, and scope.
2. Declare construction/repair participation and the full independence vector.
3. Validate archive or source-release safety and exact manifest before loading target code.
4. Extract claims, definitions, assumptions, nonclaims, and applicable standards.
5. Map claims/clauses to evidence and enforcement.
6. Run bounded decisive checks and materially distinct reconstruction where practical.
7. Attack claim binding, TCB overlap, mutations, literature, significance, provenance, packaging, and presentation.
8. Record exact methods, commands, runtimes, resource bounds, skipped checks, and uncertainty.
9. Verify target bytes/tree remained unchanged.
10. Issue target rulings, review-completion state, findings, and narrow conclusion separately.

72.3 Ruling table

Target	Scientific ruling	Basis / finding IDs	Blocks active presentation?
--------	-------------------	---------------------	-----------------------------

Create one row per headline claim/standard target plus applicable binding, reconstruction, scope, integrity, novelty, significance, and reviewability targets.

Scientific rulings only:

- PASS
- PASS -- WITHIN STATED SCOPE
- FAIL
- NOT ESTABLISHED
- NOT TESTED
- NOT APPLICABLE
- UNRESOLVED

Repair needs are findings, not rulings. Print the exact review-object hash immediately below the table.

72.4 Findings

Every actionable finding states stable ID, severity, exact location, evidence, consequence, required action, objective closure test, blocking Boolean, and successor/new-review requirement. Vague advice is nonconforming.

72.5 Prohibited

- Modify, repair, rebuild, or repackage the target.
- Call role separation independence without evidence.
- Transfer review across hashes.
- Grant human/external status.
- Invent defects or let persona dictate verdict.
- Call unperformed work passed.

72.6 Fairness

Hostility controls search intensity, not outcome. Concede every surviving result or clause at its exact demonstrated scope.

73 Professor Grumpy adversarial-AI project profile

- Status: active ACGR project profile included with ACGR-M v3.0 Release Candidate 1
- Base role: `ROLE_SEPARATED_HOSTILE_REVIEWER_ROLE.md`
- Authority: direct user mandate

73.1 Exact identification

- Review filename ends `_GRUMPYAI.md`.
- Near the top:

****Review type:**** Adversarial AI red-team review

****Reviewer:**** Professor Grumpy

****Signature:**** GRUMPYAI

****Human peer review:**** No

- The report ends with `GRUMPYAI`.

73.2 Personality prompt

You are **Professor Grumpy**, ACGR's merciless adversarial AI referee. You have no invented university, biography, romantic vendetta, or human credential. Your authority comes only from the precision, reproducibility, and fairness of your attacks.

Your obsession is destroying AI-slop science. Fluent filler, confidence theater, decorative equations, green checkmarks, self-awarded independence, enormous logs, and polished PDFs receive no evidential credit. If prose says "obviously," look for a counterexample. If metadata says "verified," ask what was recomputed. If validators agree, identify shared bugs. If a package praises itself, check whether it wrote the applause into expected output.

Be maximally adversarial and difficult to impress. Nit-pick every detail capable of exposing scientific or assurance failure: quantifiers, signs, edge cases, citations, author names, stale fields, filenames, runtimes, missing nonclaims, exclusions, imports, mutation resistance, typography, and status wording.

Humor is dry and aimed at bad scientific habits, never people. A circular validator may be "a mirror giving itself a peer-review badge"; failed novelty search may be "priority by the ancient method of not looking hard enough." Humor is never evidence.

Never invent a defect, exaggerate severity, move a met burden, alter the target, imply human review, or let tone determine a verdict. Concede what survives at exact scope. Record what you did not check. Give specific, objectively closable recommendations.

Follow the base role and applicable standard exactly.

74 ACGR-M v3.0 Standards Index

- Status: normative index for ACGR-M v3.0 Release Candidate 1; adoption remains gated by review and explicit release authorization

ID	File	Scope	Applicability lead
QS-01	QS-01_SCIENTIFIC_INTEGRITY_EVIDENCE_AND_LITERATURE.md	Claims, evidence, TCB, literature, novelty, significance	All consequential scientific work
QS-02	QS-02_COMPUTATION_OPERATIONAL_ASSURANCE_AND_CONTINUITY.md	Compute authorization, capability safety, runtime, split closure, continuity	Cost/risk/profile triggered
QS-03	QS-03_PROMOTION_ROLE_SEPARATED_HOSTILE_REVIEW_AND_SUCCESORS.md	Promotion lifecycle, hostile review, findings, successors	Review-release candidates
QS-04	QS-04_REVIEWER_ARTIFACT_AND_PRESENTATION_QUALITY.md	Immutable reviewer object, reviewer time, visual/structured QA	Reviewer packages
QS-05	QS-05_EXTERNAL_REVIEWER_AND_PUBLIC_STATUS_GOVERNANCE.md	Human qualification, conflicts, aggregation, public tiers	Only when external/human/public functions activate

ID	File	Scope	Applicability lead
QS-06	QS-06_EXTERNAL_SOURCES_REPRODUCIBILITY_APPLICATION_AND_MEASUREMENT.md	Untrusted sources, authenticity, reproducibility, applications, energy	Triggered by relevant claims/inputs

Each adopted standard must have stable clause IDs, canonical Markdown, version/hash, applicability, owner, evidence class, non-waivable rules, failure actions, schemas, fixtures, migration, and v2.7 mapping. Profiles may strengthen but cannot silently weaken exactness, evidence preservation, immutable history, truthful review classification, or claim scope.

75 QS-01 — Scientific Integrity, Evidence, and Literature

- Status: normative component of ACGR-M v3.0 Release Candidate 1; adoption remains gated by review and explicit release authorization
- V2.7 relationship: bonifies claim characterization, claim registry, exact acceptance, TCB, assurance, negative knowledge, finite claims, frontier provenance, and A2/A3 gates

75.1 Required independent judgments

- correctness;
- exact scope and applicability;
- novelty/frontier status;
- scientific significance;
- reviewability;
- evidence integrity and provenance;
- independence grade.

No overall PASS may conceal an unresolved axis.

75.2 Claim and evidence gate

Every consequential claim exposes:

- stable claim ID and exact text;
- definitions, quantifiers, assumptions, domain, exclusions, and nonclaims;
- evidence class: proof, exact finite certification, empirical support, bounded negative, counterexample, heuristic, or unresolved;
- raw evidence and transformation chain;

- imported theorem dependencies and exact mappings;
- acceptance semantics and exact claim-binding route;
- residuals, counterexamples, known limitations, and retry conditions.

Raw types, shapes, alphabets, labels, encodings, and metadata are validated before normalization. A correct object relabeled for a broader or different claim must fail even if all hashes are recomputed.

75.3 TCB, independence, and mutation gate

Every decisive validator discloses overlap in actor, code, parser, data, representation, algorithm, mathematical formulation, libraries/runtime, assumptions, and expected answers. Use only the strongest label justified: separate implementation, implementation-independent reconstruction, clean-room reconstruction, independent agent red-team, independent human review, or external replication.

At higher assurance, run mutations aimed at named failure modes: malformed shapes/types, boundary values, plausible wrong witnesses, altered claim binding, changed evidence with recomputed hashes, omitted/extra members, stale status, and false expected sentinels. Choose the smallest independent route that closes the consequential failure mode; duplicate full computation only when the claim requires it.

75.4 Literature and imported-theorem gate

For every headline or novelty component, maintain a claim-neighbor matrix covering:

- closest primary sources and stable source IDs;
- alternate terminology and search routes;
- stronger, weaker, equivalent, or incomparable relation;
- assumptions, objective, method, guarantee, evidence scale, chronology, and domain;
- exact effect on wording, nonclaims, significance, or promotion;
- unresolved search limits and review date.

Dangerous sources capable of subsuming or narrowing the claim must appear in the reviewer-facing comparison. Failed retrieval establishes non-detection under a recorded search, not novelty or priority.

Imported theorems record exact source/version, theorem identifier, assumptions, conclusion, campaign mapping, and proof of any scope extension. Stronger, contradictory, or superseding internal results must be incorporated, excluded with precise reason, or quarantined.

75.5 Significance gate

State the recognized question affected, proof obligation reduced, family/case closed, reusable structure or method created, and downstream action enabled. Correct but consequence-poor results remain certified and searchable outside the scarce active-promotion surface.

Finite examples do not establish prevalence. Formal or symbolic results do not establish systems impact. Evidence cannot move upward from mechanism existence to frequency, robustness, scalability, application, or physical impact without new evidence.

75.6 Claim-surface consistency

Before promotion compare claim registry, raw evidence, proof, quick-read, literature matrix, limitations, manifest, promotion record, catalogs, and public status. Contradictory or withdrawn quantities, scope, identities, citations, runtime statements, or status language block release.

75.7 Mechanical versus attested judgment

Mechanical validators may verify file presence, schemas, hashes, identities, exact recomputation, declared dependencies, and transition authority. They may not establish novelty, significance, reviewer expertise, conflict freedom, external validation, or scientific interpretation.

Judgment records require identified actor, authority, evidence, scope, date, uncertainty, and signature/provenance. A complete form is not proof. Missing or unsigned required attestation leaves the corresponding state **UNRESOLVED**.

Every schema constraint, diagnostic, and fixture cites canonical clause IDs. Orphan constraints and normative clauses with no implementation or attestation route fail conformance.

75.8 Non-waivable core

- exact claim/evidence binding;
- raw-evidence validation;
- truthful evidence class and scope;
- preservation of counterexamples and negative knowledge;
- no novelty from non-retrieval;
- no external/human independence without evidence.

76 QS-02 — Computation, Operational Assurance, and Continuity

- Status: normative component of ACGR-M v3.0 Release Candidate 1; adoption remains gated by review and explicit release authorization
- V2.7 relationship: bonifies generation contracts, information efficiency, launch/runtime/checkpoints, handoff, transfer, efficiency review, and autonomy

76.1 Cost applicability

A profile defines thresholds across predicted wall time, CPU-hours, peak memory, storage/output, restart cost, irreversibility, and scientific consequence. Crossing any threshold activates the corre-

sponding long-run obligations. Unknown or unmeasured cost triggers a bounded scaling preflight. The exact small-domain correctness gate is non-waivable regardless of cost.

76.2 Long-run authorization

Freeze exact question, discriminated hypotheses, full-semantic pilot, oracle agreement, representation capability, semantics-preserving reductions, complete-pipeline scaling, information gain, theorem-obligation reduction, resource ceilings, process ownership, checkpoint/recovery, and stop/saturation/pivot rules.

Use frontier reasoning for formalization/reduction/design; deterministic hardware for validated bulk work. Search symmetry, canonicalization, dominance, dynamic programming, memoization, incremental recomputation, pruning, parallel decomposition, and compact certificates before scale.

76.3 Capability-safe operator surface

`status`, `self-test`, `install`, `run`, `resume`, `handoff`, and `transfer` are separate capabilities. Status and self-test **MUST be technically incapable** of canonical scientific mutation or launch. Missing, malformed, or dropped capability/mode arguments **MUST fail closed**.

Authorization occurs last after exact bytes and real operator path pass. Terminal work cannot be resumed by ordinary launchers.

76.4 Live observability

Long runs expose process-tree identity, lock ownership, heartbeat, durable progress, resources, outstanding obligations, checkpoint health, expected completion, and safe next action. Graceful stop, bounded child termination, orphan scan, crash capture, checkpoint validation, and restart decision are distinct.

Optional deep checks declare runtime, progress, ceiling, information value, and stop guidance. Stopped or unperformed work is not passed work.

76.5 Split closure

- `TERMINAL_Scientific`: accepted evidence, runtime, claims, negative knowledge, incidents, state, and locks are scientifically closed.
- `HANDOFF_CLOSED`: the scientific handoff is complete and validated.
- `TRANSFER_CLOSED`: the continuity transfer is complete and validated.

Scientific terminality does not depend on successful packaging. Handoff/transfer failure blocks release or continuity, remains preserved as operational failure, and never rewrites valid scientific state.

76.6 Platform/durability

Declare local disk, Windows replacement, synchronized folder, network storage, and other material assumptions. Require unique staging, verified copy/promotion, post-change fingerprints, quarantine, and fresh gates after tree changes.

76.7 Required fixtures

- unknown-cost preflight;
- forced interruption/recovery;
- dropped self-test/status argument unable to run/mutate;
- scientific terminal plus handoff/transfer failure;
- synchronized-storage checkpoint/reappearance;
- optional deep check stopped honestly;
- output materialization defeating claimed complexity.

76.8 Non-waivable

Small exact gate, fail-closed capabilities, truthful resource authorization, scientific-state preservation, and split closure semantics.

77 QS-03 — Promotion, Role-Separated Hostile Review, and Successors

- Status: normative component of ACGR-M v3.0 Release Candidate 1; adoption remains gated by review and explicit release authorization
- Specialty lead: Hadamard promotion/review history
- Direct mandate: extreme adversarial review is central and non-negotiable

77.1 Lifecycle

```
WORKING_CLAIM
-> INTERNALLY_SUPPORTED
-> INTERNAL_NONCERTIFYING_PREFLIGHT
-> REVIEW_RELEASE_CANDIDATE
-> CANDIDATE_REVIEW_OBJECT_FROZEN
-> ROLE_SEPARATED_ADVERSARIAL_AI_REVIEW
-> FINDING_DISPOSITION
-> ACTIVE_REVIEW_SURFACE | SUCCESSOR_REQUIRED | WITHDRAWN
```

Review cannot precede immutable bytes. Internal preflight is not hostile-review certification. Reviews remain external attestations bound to object hashes.

77.2 Independence

Role separation is mandatory; independence is graded and never self-awarded. Review records disclose construction/repair participation, actor/model/session, shared context/expected answers, code/data/parser/representation/formulation/runtime/TCB overlap, selection authority, and organizational control.

77.3 Separate dimensions

- Scientific ruling per target.
- Review completion.
- Finding severity and closure.
- Presentation action.

Scientific rulings: PASS, PASS_WITHIN_SCOPE, FAIL, NOT_ESTABLISHED, NOT_TESTED, NOT_APPLICABLE, UNRESOLVED. There is no PASS_REPAIRS_REQUIRED.

77.4 Mandatory ruling and findings tables

Target	Scientific ruling	Basis / finding IDs	Blocks active presentation?
--------	-------------------	---------------------	-----------------------------

Rows cover every headline claim and applicable binding, reconstruction, scope, integrity, novelty, significance, and reviewability target. Print exact review-object type and SHA-256 below the table.

ID	Severity	Exact target/location	Evidence	Consequence	Required action	Closure test	Blocks?
----	----------	-----------------------	----------	-------------	-----------------	--------------	---------

Recommendations name operational action and objective closure evidence. Vague "clarify/improve/clean up" advice fails.

77.5 Blocking predicate

Active presentation requires frozen object, required review completion, all applicable promotion axes satisfied, and no open fatal or interpretation-changing blocking-major finding. Such findings cannot be waived. They may be fixed, disputed with evidence, or reclassified by authorized signed disposition.

77.6 Successors

Every successor records predecessor hash, finding IDs, dispositions, changed paths/hashes, successor type, scientific-payload map, closure evidence, and residual limits. Presentation-only successors cannot claim scientific repair. New object hashes require explicit review inheritance analysis or new review; no silent transfer.

77.7 Universal versus project profile

This standard requires typed reviewer metadata and truthful AI/human classification. Exact Professor Grumpy naming, disclosure, suffix, and signature belong to the active project profile under roles/.

77.8 Required fixtures

- review before frozen object;
- builder/repairer claiming independence;
- ruling passing while blocking major remains open;
- AI review populating human state;
- typography-only successor claiming scientific repair;
- unsupported prose surviving evidence removal;
- verdict silently transferred across hash.

78 QS-04 — Reviewer Artifact and Presentation Quality

- Status: normative component of ACGR-M v3.0 Release Candidate 1; adoption remains gated by review and explicit release authorization
- Applicability: reviewer-package engineering and presentation assurance; no external reviewer qualification/public ranking rules

78.1 Immutable review object

The default scientific object is one immutable ZIP with one top-level directory and minimal sufficient closure: quick-read/source, exact claims/nonclaims, literature/significance, evidence/proof objects, production source/frozen inputs, distinct validator, decisive command, deep routes, environment/runtime/platform, manifests/hashes/provenance/TCB/limitations/status/lineage, and material companions.

Protocol/source releases use a registered immutable review-object type. The external hash closes complete bytes. Target metadata and payload manifest declare members and external review-output exclusion.

78.2 Reviewer time

Within minutes, expose exact claim/nonclaims, strongest vulnerability, closest prior work, significance, first command, scope, runtime, failure meaning, and deep evidence. Brevity cannot remove assumptions or diagnostics; decorative bulk and project archaeology are defects.

78.3 Closed-world and non-mutation

Inspect safety; establish exact file set; hash all members; reject extras/omissions; disable/externalize caches; then load target code; validate raw evidence; recompute claims; mutate; verify extracted tree unchanged. Fail nonzero with exact diagnostics.

78.4 Presentation QA

Clean-build; check fonts/glyphs/text; render every page; inspect clipping, overlap, equations, tables, links, hierarchy, margins, and page flow; compare displayed facts to canonical evidence; test commands/runtimes; preserve inspection record; hash-bind final bytes.

Spreadsheets require schema/formula/link/filter integrity, row-to-source spot checks, duplicate detection, and visual inspection. Rebuild uncertain artifacts from immutable predecessors.

78.5 Identity

Validate authors, citations, status, versions, hashes, and reviewer classes. Never invent affiliations. Personal author lines remain profiles.

78.6 Required fixtures

- self-mutating validator;
- extra/missing member;
- stale claim/metadata/citation identity;
- unreadable but mathematically correct PDF;
- missing material companion;
- command/runtime mismatch;
- source-release review object with changed target file.

79 QS-05 — External Reviewer and Public-Status Governance

- Status: conditionally applicable normative component of ACGR-M v3.0 Release Candidate 1; adoption remains gated by review and explicit release authorization
- Applicability: inactive unless external/human review or public governance is invoked
- Rule: must not leak A3/public-portal burden into internal A0/A1 or ordinary A2 mathematics

79.1 Qualification authority

The project owner or explicitly delegated Review Qualification Authority verifies bounded reviewer status. A supervisor may prepare records but cannot self-approve qualification involving its own review. Qualification verifies evidence and scope; it does not guarantee a verdict.

79.2 Reviewer record

Record:

- identity anchor or verified-private-anonymity record;
- bounded subject expertise and supporting publications/software/formalization/datasets;

- effective and expiry/recheck dates;
- artifact-specific conflict decision;
- organizational/review dependency graph;
- approved review scope;
- review-quality requirements;
- verifier and decision authority;
- appeal/dispute trail.

Conflict review covers authorship, project contribution, institution/group, recent collaboration, financial interest, professional dispute, and method involvement. Expired qualification cannot silently retain current status.

79.3 Review quality

Qualified reviews identify exact object hash, claims/clauses, methods, commands, evidence, limitations, conflicts, and reasoned dispositions. Prestige and positivity are not quality. Verified anonymity may protect public identity while retaining private evidence.

Unqualified criticism remains visible but cannot automatically alter qualified-review state.

79.4 Aggregation

Do not average reviews or count dependent reviews as independent. Cluster by shared actor, institution, method, data, code, and organizational control. Preserve positive, negative, mixed, contested, withdrawn, and expired statuses separately.

79.5 Public tiers

1. Certified result — not human expert-reviewed.
2. Active review candidate — adversarial AI reviewed.
3. Qualified human review within stated scope.
4. Multiple materially independent qualified reviews or external replication.
5. Mixed or contested.
6. Suspended or withdrawn.

Every public item displays exact version/hash, claim/scope, novelty, significance, AI-review status, human-review count/scope/dependency, unresolved blockers, nonclaims, and verifier command. Negative and mixed reviews cause visible status consequences and remain linked with responses/successors.

Default human label is **NOT EXPERT-REVIEWED**. AI review never increments human counts. Publication authority remains separate from conformance.

79.6 Appeals

Appeals are append-only, evidence-bearing, and adjudicated by the project owner or separately designated authority. They cannot delete the original review. Revised qualification or finding status creates a new event with rationale.

79.7 Required fixtures

Positive, negative, mixed, dependent, expired, anonymous-verified, conflicted, appealed, and unqualified reviews; AI review attempting human status; public ordering with visible negative demotion.

80 QS-06 — External Sources, Reproducibility, Application, and Measurement

- Status: normative component of ACGR-M v3.0 Release Candidate 1; adoption remains gated by review and explicit release authorization
- Specialty leads: Hadamard external-source certification and EOLSD application/measurement assurance

80.1 Untrusted external-source ingestion

Treat scripts, notebooks, encoded posts, binaries, model outputs, and executable-looking scientific sources as hostile by default.

- Preserve original bytes, retrieval URL/description, method, time, and hash.
- Do not execute by default.
- Prefer static, non-shell, allowlisted decoding or safe reimplementations.
- Hash-link every transformation stage.
- If execution is unavoidable, require explicit authority, isolation, resource limits, and network-disabled operation where practical.
- Bind the final literal witness independently to the claim.
- Record mirrors, signatures, or third-party anchors where available.

Separate local capture integrity, provenance continuity, external authenticity, and scientific correctness. Self-hashes prove only local byte identity.

80.2 Reproducibility classes

Declare each material output as:

- bitwise deterministic;
- seed-reproducible;
- semantically reproducible;
- stochastic with exact acceptance;
- platform-bounded.

Bitwise claims control encoding, newlines, numeric representation, ordering, locale, paths, timestamps, archive metadata, and software versions. Semantic claims define parser, schema, and equivalence relation. Test every claimed platform or narrow the claim honestly.

80.3 Formalization and application bridge

Map external/literature definitions to internal representations and verifier semantics. For applied claims, a formal surrogate remains model-internal until the mapping to the target system/domain is derived or empirically validated.

An application bridge states:

- target object/system;
- formal surrogate;
- mapping assumptions and failure modes;
- derivation or validation evidence;
- domain and uncertainty;
- exact conclusions allowed and prohibited.

80.4 Systems and empirical evidence scale

Distinguish mechanism existence, prevalence, effect size, robustness, scalability, system impact, and physical impact. Evidence at one rung does not establish a higher rung. Systems claims require realistic workloads, strong baselines, ablations, overhead-inclusive end-to-end measurements, uncertainty, and negative cases appropriate to scope.

80.5 Environmental and energy claims

Operation counts, symbolic costs, controller time, or model-internal proxies are not physical energy evidence. Physical claims require:

- defined useful-output unit;
- power integration and joules per unit;
- CPU/GPU/RAM and system boundary;

- hardware/software configuration;
- baselines, workload, run conditions, repetitions, and uncertainty;
- separation of algorithmic savings from measured system energy.

Without that evidence, use literal operation-count or model-cost language only.

80.6 Non-waivable core

- no execution of untrusted material by convenience;
- no authenticity claim from self-hashes;
- no bitwise claim without controlled serialization;
- no applied or environmental inference without the corresponding bridge and measurements.

81 Version 2.7 Preservation and Clause Map

- Baseline TeX bytes: 145,582
- Baseline TeX SHA-256: A4FC77434041C8D74914CA370145B413642996154CAECF7D36AE37223AF0CA82
- Baseline PDF SHA-256: 2C67C68AD5C4A2BC079BAD996AB42CBC5FE9E7979EECF128BA67C7E3B91669D1
- Default disposition: RETAINED
- Retirement count: 0

The Version 3.0 core began as an exact byte copy of the Version 2.7 TeX. Every Version 2.7 section, paragraph, list item, example, template, and invariant remains in the Version 3.0 source unless this map explicitly states otherwise. Version 3.0 adds metadata, version-facing bootstrap wording, and a generated governance/quality-system layer. It does not delete a Version 2.7 scientific obligation.

81.1 Section roots

V27-S01 Version 2.7 section: Core Principle

Disposition: RETAINED_AND_BONIFIED

Version 3.0 treatment: Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.

V27-S02 Version 2.7 section: Version Lineage: What Version 2.3 Established and Version 2.7 Preserves

Disposition: RETAINED_AND_BONIFIED

Version 3.0 treatment: Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.

V27-S03 Version 2.7 section: Normative Language**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S04 Version 2.7 section:** What the Agent Is Asked to Do**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S05 Version 2.7 section:** Cold-Start Orientation for a Fresh or Context-Poor Agent**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S06 Version 2.7 section:** Suitability Check Before Search**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S07 Version 2.7 section:** Three-Layer Research Model**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S08 Version 2.7 section:** Mathematical and Campaign State**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S09 Version 2.7 section:** Claim Characterization and Canonical Claim/Assurance Record**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S10 Version 2.7 section:** Claim Registry and Evidence Dependency Closure**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.

V27-S11 Version 2.7 section: Claim-Bound Exact Acceptance**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S12 Version 2.7 section:** Trusted Computing Base and Verifier Independence**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S13 Version 2.7 section:** Consequence-Sensitive Assurance and Gate Closure**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S14 Version 2.7 section:** Counterexamples, Residuals, and Negative Knowledge**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S15 Version 2.7 section:** The Mature ACGR-M Loop**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; authority, roles, evidence vocabulary, and applicability are bonified by the v3 governance layer.**V27-S16 Version 2.7 section:** Scientific Generation Contract**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S17 Version 2.7 section:** Representation Pressure**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S18 Version 2.7 section:** Discovery, Compression, and Proof**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.

V27-S19 Version 2.7 section: Compression Soundness and Coverage**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S20 Version 2.7 section:** Targeted Challenge Generation and Novelty**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S21 Version 2.7 section:** Adaptive Learning and Counterfactual Utility**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S22 Version 2.7 section:** Exact Acceptance and Verifier Diversity**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S23 Version 2.7 section:** Verifier/Oracle Incidents, Quarantine, and Repair**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S24 Version 2.7 section:** Numerical and Objective Discipline**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S25 Version 2.7 section:** Regression Protection and Semantic Equivalence**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S26 Version 2.7 section:** Reproducibility Classes and Controlled Nondeterminism**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.

V27-S27 Version 2.7 section: Finite Exhaustion, Minimality, and Negative Claims**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S28 Version 2.7 section:** Generations, Mechanisms, and Campaigns**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S29 Version 2.7 section:** Stop Signals and Handoff Criteria**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S30 Version 2.7 section:** Benchmarking and Information Efficiency**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S31 Version 2.7 section:** Exploratory Discovery and Confirmatory Evaluation**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S32 Version 2.7 section:** Benchmark Governance and Contamination**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S33 Version 2.7 section:** Ablations and Causal Attribution**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S34 Version 2.7 section:** Generation Ledger and Immutable Lineage**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.

V27-S35 Version 2.7 section: One Implementation per Generation**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; information value, theorem obligations, compression debt, and search gates are bonified by QS-01/QS-02.**V27-S36 Version 2.7 section:** Human-Lightweight Project Layout**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.**V27-S37 Version 2.7 section:** Project Identity, Hash Pinning, and State Manifest**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.**V27-S38 Version 2.7 section:** Schema Versioning and Migration Discipline**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.**V27-S39 Version 2.7 section:** Safe Launch and Runtime Invariants**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.**V27-S40 Version 2.7 section:** Runtime Accounting**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.**V27-S41 Version 2.7 section:** Transactional Checkpoints, Worker Isolation, and Commit Discipline**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.**V27-S42 Version 2.7 section:** Forced Interruption and Recovery Testing**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.

V27-S43 Version 2.7 section: Release-Artifact and Operator-Surface Closure**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.**V27-S44 Version 2.7 section:** Automatic Handoff Package**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.**V27-S45 Version 2.7 section:** Agent-Continuity Transfer ZIP**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; capability safety, live status, process ownership, platform profiles, and split closure are bonified by QS-02.**V27-S46 Version 2.7 section:** Mathematical-Result Promotion and Skeptical Human Audit**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S47 Version 2.7 section:** Evidence Retention, Compaction, and Cold Storage**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S48 Version 2.7 section:** What the Refining Agent Must Do With the Handoff**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S49 Version 2.7 section:** Mandatory Efficiency Review**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S50 Version 2.7 section:** Resource Allocation Across Research Lanes**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.

V27-S51 Version 2.7 section: Branches, Competing Representations, and Merge Discipline**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S52 Version 2.7 section:** Generation Release Gate – A1 Detailed Acceptance**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S53 Version 2.7 section:** Claim Promotion Gate – A2**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S54 Version 2.7 section:** Independent Audit Gate – A2/A3**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S55 Version 2.7 section:** Publication and Novelty Gate – A3**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S56 Version 2.7 section:** Independent Reconstruction and Clean-Room Replication**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S57 Version 2.7 section:** Proof Red-Team Phase**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S58 Version 2.7 section:** External Frontier Provenance**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.

V27-S59 Version 2.7 section: Protocol Evolution Across Campaigns**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; promotion, hostile review, findings, successor repair, reviewer classes, and public status are bonified by QS-03/QS-05.**V27-S60 Version 2.7 section:** Human's Entire Job**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; role architecture, continuity, source authority, and conformance routing are bonified by the v3 role/governance files.**V27-S61 Version 2.7 section:** Agent Autonomy Contract**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; role architecture, continuity, source authority, and conformance routing are bonified by the v3 role/governance files.**V27-S62 Version 2.7 section:** Optional Proof-Complete Branch**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; role architecture, continuity, source authority, and conformance routing are bonified by the v3 role/governance files.**V27-S63 Version 2.7 section:** Minimal Machine-Readable Refinement Signal**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; role architecture, continuity, source authority, and conformance routing are bonified by the v3 role/governance files.**V27-S64 Version 2.7 section:** Recommended Machine-Readable Generation Entry**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; role architecture, continuity, source authority, and conformance routing are bonified by the v3 role/governance files.**V27-S65 Version 2.7 section:** Final Instruction Template**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; role architecture, continuity, source authority, and conformance routing are bonified by the v3 role/governance files.**V27-S66 Version 2.7 section:** Summary**Disposition:** RETAINED_AND_BONIFIED**Version 3.0 treatment:** Retained; role architecture, continuity, source authority, and conformance routing are bonified by the v3 role/governance files.

81.2 Clause-level preservation invariant

Every material Version 2.7 obligation inherits the stable prefix of its containing section root and remains authoritative. Paragraph- and list-level machine identifiers are derived as `V27-Snn-Lmmmm-Hhhhhhhhhhhhh`, where `Lmmmm` is the frozen baseline source line and `H...` is the first 12 hexadecimal characters of the normalized UTF-8 clause SHA-256. The machine registry is `governance/V2_7_CLAUSE_REGISTRY.json`.

A future edit that removes, weakens, moves, or supersedes inherited text must update this map and the registry in the same successor release. `RETIRED_WITH_AUTHORIZATION` requires direct authority and regression evidence. Absence from a generated view is a conformance failure, not implicit retirement.

82 Changelog from ACGR-M v2.7 to v3.0

82.1 Preserved foundation

The complete 145,582-byte Version 2.7 TeX source was copied into the Version 3.0 core before editing. Its scientific research engine, examples, rationale, assurance architecture, operational disciplines, and transfer-continuity guidance remain present. No Version 2.7 section was retired.

Frozen Version 2.7 identities:

- TeX SHA-256: `A4FC77434041C8D74914CA370145B413642996154CAECF7D36AE37223AF0CA82`
- PDF SHA-256: `2C67C68AD5C4A2BC079BAD996AB42CBC5FE9E7979EECF128BA67C7E3B91669D1`

82.2 Bonifications

Version 3.0 adds:

1. separate supervisor, researcher, and role-separated hostile-reviewer contracts;
2. a project-specific Professor Grumpy adversarial-AI profile without misrepresenting it as human peer review;
3. immutable review objects and exact review-target identity;
4. separate scientific ruling, review-completion, finding, presentation, and public-status dimensions;
5. explicit blocking predicates, actor permissions, forbidden states, and fail-closed transitions;
6. six quality standards covering science, computation, hostile review, reviewer artifacts, external review/public status, and external sources/application/measurement;
7. profile-level computation thresholds, bounded scaling preflight, interruption/recovery tests, and split scientific/handoff/transfer closure;
8. external human-review qualification, conflict, dependency, expiry, appeal, and public-tier governance;

9. a canonical-source graph, deterministic generated views, manifests, and clause-preservation records;
10. author/contact/repository metadata, a public identity fingerprint, and exact release hashes.

82.3 Deliberately not changed

- Exactness is not tiered.
- Finite evidence is not promoted to a universal theorem.
- Search failure is not nonexistence without certified completeness.
- Hashes do not prove truth, novelty, authorship, expertise, or independence.
- Reviewer hostility does not permit invented defects or personal attacks.
- Public status and human review cannot be manufactured mechanically.

83 Migration from ACGR-M v2.7

Version 3.0 applies prospectively unless an artifact owner explicitly migrates an existing campaign. Existing Version 2.7 evidence remains valid at its demonstrated scope; it is not silently upgraded to Version 3.0 conformance.

83.1 New campaign

Use the Version 3.0 core, declare applicable QS profiles, assign the three roles, register canonical sources and state axes, and freeze exact review objects before adversarial review.

83.2 Existing campaign

1. Preserve the original Version 2.7 state and artifacts unchanged.
2. Record the migration event, owner, authority, source and destination versions, and exact hashes.
3. Map existing claims, evidence, roles, review records, promotion state, handoff state, and transfer state onto the Version 3.0 axes.
4. Leave unknown or unsupported fields unresolved; do not infer stronger status from filenames or old generic PASS labels.
5. Apply only the standards triggered by the campaign's actual scope and profile.
6. Build new fixtures for any materially changed parser, verifier, schema, operator path, or claim-binding rule.
7. Freeze a successor review object and route it through role-separated hostile review before claiming Version 3.0 review status.

83.3 Grandfathering

Grandfathering may preserve historical operation but cannot waive exact acceptance, claim-evidence binding, evidence boundaries, immutable provenance, non-fabrication, fatal blockers, or interpretation-changing major blockers. A grandfathered exception must be explicit, scoped, authorized, dated, and visible wherever it affects interpretation.

84 Summary

ACGR-M v3.0 is deliberately full-context. It preserves the entire Version 2.7 scientific engine because that detail is part of the decision system, not ornamental exposition, and adds explicit role, review, governance, computation, presentation, and release controls without collapsing the older research guidance.

Its mature loop is:

**Freeze claim → represent → search → validate raw/exactly → bind
evidence to claim → preserve residuals → learn → challenge → confirm →
compress/certify coverage → challenge the representation → attempt proof
→ red-team/reconstruct → promote selectively → assess transition →
refine.**

Its central invariants are:

1. **No accepted mathematics without exact evidence, raw-domain validation, and explicit claim-evidence binding.**
2. **Finite computation is never silently promoted to a universal theorem.**
3. **Nonexistence, exhaustiveness, and minimality require certified completeness.**
4. **Failures and dead ends are preserved as scientific evidence.**
5. **Representations are hypotheses and must themselves be challenged.**
6. **Every generation must increase mathematical capability, information efficiency, verification strength, or justified understanding – not merely runtime.**
7. **Optimize aggressively, but never trade mathematical quality for speed.**
8. **One frozen implementation per generation; complete append-only lineage across the campaign.**
9. **The human is the custodian. The agent is responsible for routine mathematics and research engineering.**
10. **Heuristic objective improvement is not automatically mathematical progress or distance to proof.**
11. **Accepted claims retain auditable dependency paths to assumptions, evidence, verifier semantics, scope, and claim-evidence mapping.**

12. The exact packaged release and actual operator path, not merely the development tree, must pass the release gate.
13. Parallelism and nondeterminism may change discovery order, never exact acceptance semantics or uncertified completeness.
14. Verifier independence is a claim about shared trust boundaries, not merely two executable filenames.
15. Exploratory evidence may select a mechanism; it must not masquerade as unbiased confirmation.
16. Compression does not replace evidence until its claimed coverage or equivalence is exactly established.
17. A known-bad oracle or claim-binding dependency quarantines downstream claims until they are re-established.
18. Exactness is never tiered. Surrounding assurance scales with consequence and is applicability-closed.
19. Generic PASS labels never hide independent assurance dimensions.
20. Integrity, provenance continuity, and authenticity are distinct trust claims.
21. A generation may advance the research without producing a promotable theorem.
22. Promoted computation-dependent claims must be human-auditable and reconstructible from the minimal sufficient evidence closure.
23. Correctness, novelty/frontier status, and scope are reported separately.
24. Terminal states are sealed; long computation is observable, recoverable, and safely cancellable.
25. Schemas and migrations are versioned and non-destructive; changes in meaning trigger scientific lineage and re-verification.
26. Assurance work must close a named failure mode; duplicate computation is not automatically independent assurance.
27. Protocol evolution is conservative, but protocol compression must not delete decision-critical guidance.
28. A fresh agent must read the full protocol and campaign transfer brief before making material successor decisions.
29. Every AI-mediated A1+ generation provides continuity-transfer tooling: the hand-off preserves authoritative science, while the transfer ZIP preserves the context required to continue that science correctly.

The deepest objective is not to become an ever-faster enumerator or an ever-more elaborate assurance bureaucracy. It is to make computation progressively unnecessary by converting informative failures and exact successes into structure, structure into lemmas, and lemmas into proof – while preserving enough explicit context that a new competent agent can continue the campaign intelligently rather than rediscovering its history.

HORTRAME Research

AI-use disclosure

Adaptive Counterexample-Guided Refinement for Mathematics (ACGR-M) — version 3.0

Disclosure-only publication addendum — 4 October 2026

AI-use disclosure: This work was developed with assistance from AI agents operating within the HORTRAME Systems research architecture. Depending on the research stage, these agents contributed to literature analysis, mathematical formulation, proof reconstruction, adversarial review, counterexample search, simulation and code analysis, verification design, reproducibility checks, revision, and manuscript preparation. AI-agent outputs were treated as research artifacts subject to verification and correction rather than as independent authority. G. Bolduc is the human author responsible for the manuscript, its claims, source selection, and publication decisions. AI-agent review does not constitute independent human peer review, proof-assistant certification, experimental validation, or publication acceptance.

The preceding manuscript pages are retained unchanged. This addendum standardizes disclosure; it supplies no new proof, experiment, independent human review or acceptance. Earlier source attributions remain historical provenance, not AI authorship of this publication edition.