PROOF_LOOPS.md

Field Notes on Machines That Turn Work Into Capability

A research architecture for proof-gated autonomous work, Chronicle memory, and on-chain graph-root commitments

Vincent Boucher

President, QUEBEC.AI & MONTREAL.AI

Research Edition v2.0 | Publication-safe, claim-bounded

Core loop

Objective → Proof Mission → AGI Jobs → Evidence → Chronicle → Validated Skill → On-chain Graph Root → Future Mission Improvement

AI creates output. GoalOS creates proof. The machine may draft; the institution must decide what earns memory.

Abstract

PROOF_LOOPS.md specifies a compact operating law for proof-gated autonomous work: an objective becomes a proof mission; proof debt becomes AGI Jobs; jobs return evidence; evidence is gated by Chronicle; validated skills become graph memory; graph roots can be committed on-chain; and admitted skills improve future missions. This paper converts that operating note into a formal research architecture. It defines a state model, proof-debt mechanics, AGI Job contracts, Evidence Dockets, Chronicle admission, Validated Skill Graph accumulation, proof-level policy, on-chain commitment boundaries, and falsification tests. The central claim is architectural and testable: durable autonomy requires proof-carrying loops that prevent unverified output from becoming institutional memory. The paper does not claim achieved AGI, achieved ASI, legal certification, financial advice, production authorization, live settlement, or external validation. It defines the loop required for such claims to become inspectable, replayable, and governed.

Keywords: GoalOS; proof loops; autonomous agents; Proof Debt; AGI Jobs; Evidence Dockets; Chronicle; Validated Skill Graph; on-chain commitments; proof economy.

EXECUTIVE THESIS

A prompt is a request. A proof loop is an institution. The prompt asks a model to answer; the loop decides what must be proven before the answer can matter.

1 INTRODUCTION

Frontier models make candidate work abundant. They produce claims, reports, code, plans, critiques, tool traces, and agent outputs at low marginal cost. The institutional problem is not whether an AI system can generate. It is whether a result deserves trust, reuse, payment, propagation, memory, or rollback authority.

GOALOS answers by separating output from authority. Model output is candidate work. Candidate work may create Proof Debt. Proof Debt may become AGI Jobs. Jobs may return PROOFBUNDLE artifacts. Those artifacts may update an EVIDENCE DOCKET. Only then can CHRONICLE decide whether a capability enters durable memory as a validated skill.

The note PROOF_LOOPS.md compresses the system to one loop:

$$\begin{aligned}\text{Objective} &\to \text{Proof Mission} \to \text{AGI Jobs} \to \text{Evidence} \to \text{Chronicle} \\ &\to \text{Validated Skill} \to \text{On-chain Graph Root} \to \text{Future Mission Improvement}.\end{aligned}$$

This paper gives that loop a research form: objects, transition rules, gates, proof levels, privacy boundary, threat model, and evaluation program.

2 CONTRIBUTIONS

This paper makes seven contributions.

1. **Proof-loop formalism.** A state-machine model for converting objectives into proof-gated capability.
2. **Proof Debt to AGI Job conversion.** A job factory that starts empty and creates only the proof work required by the mission.
3. **Evidence Docket model.** A structured proof room that turns artifacts into inspectable claims, risks, replay paths, and validator records.
4. **Chronicle admission law.** A memory firewall preventing unsupported output from influencing future missions.
5. **Validated Skill Graph accumulation.** A capability graph that stores scoped, replayable, evidence-backed skills rather than prompt fragments.
6. **On-chain graph-root boundary.** A public/private model in which private intelligence remains protected while proof commitments can be anchored.

7. **Falsification program.** Tests for whether proof loops outperform report-only, prompt-library, and ungated-agent baselines.

3 System Doctrine

The proof loop rests on four doctrines.

Doctrine 1: output is not authority

A model may answer; an agent may act; an institution must prove. Fluency cannot grant future mission influence.

Doctrine 2: memory is gated

No Chronicle entry means no future mission influence. Memory must be admitted, scoped, replayable, and rollbackable.

Doctrine 3: work is proof-producing

AGI Jobs are not tasks in a todo list. They are proof contracts with claim, worker, validator, acceptance tests, blocked claims, and ProofBundle return path.

Doctrine 4: public proof, private intelligence

Do not expose private skill intelligence as plaintext. Use commitments, encrypted capsules, attestations, and zero-knowledge proofs where applicable.

4 Formal State Model

Let a proof-loop run at time t have state

$$S_t = (O, M, C, D, J, P, E, V, K, R, F), \tag{1}$$

where O is the objective, M the Mission Contract, C the claim set, D the Proof Debt register, J the AGI Job queue, P the returned ProofBundles, E the Evidence Docket, V the verifier state, K the Chronicle decision state, R the on-chain graph-root record, and F the future mission prior.

A transition is

$$S_{t+1} = \tau(S_t, u_t, \pi_\ell, \rho_t), \tag{2}$$

where u_t is an operator or system action, π_ℓ is the selected proof-level policy, and ρ_t is a review, replay, validation, or risk signal.

The loop is complete when

$$\mathrm{Done}(S_t) = M \wedge D_{retired} \wedge P \wedge E \wedge V_{pass} \wedge K_{decision} \wedge B_{boundary}, \tag{3}$$

where $B_{boundary}$ asserts that claim, privacy, wallet, settlement, and scope boundaries remain intact.

5 Layered Architecture

The proof loop is a layered operating system. Each layer narrows authority.

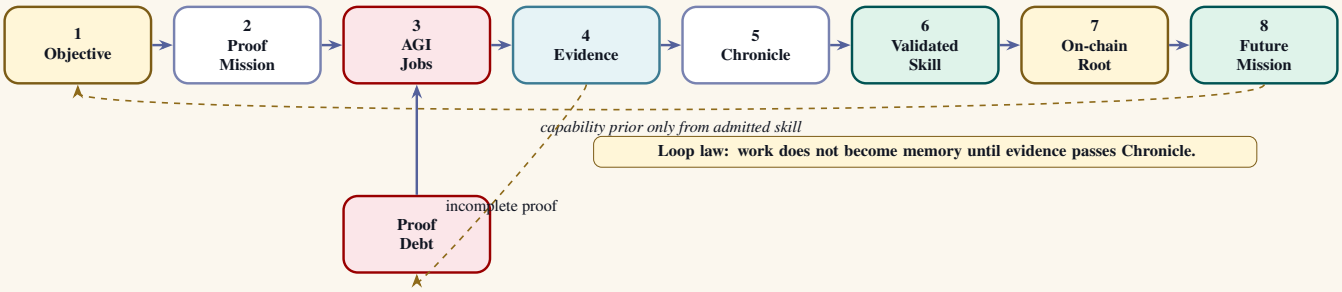

Figure 1: *The canonical proof loop. The loop turns autonomous work into proof, proof into validated skill, skill into graph memory, and graph memory into future mission improvement.*

1. Objective Layer
User intent, decision to support, success criteria, failure criteria, constraints.

2. Mission Layer
Mission Contract, proof level, claim boundary, allowed tools, validators, rollback obligations.

3. Proof Layer
Claim extraction, Proof Debt, custom AGI Jobs, worker/validator/sentinel assignments.

4. Evidence Layer
ProofBundles, Evidence Docket, replay manifests, risk ledger, source provenance.

5. Memory Layer
Chronicle Gate, validated skill admission, Validated Skill Graph, future mission influence.

6. Commitment Layer
On-chain graph roots, evidence roots, ProofBundle roots, policy hashes, attestations, encrypted capsule commitments.

Authority increases only downward: objective to proof, proof to memory, memory to commitment. A failed gate returns the run to Proof Debt.

Figure 2: *Layered proof-loop architecture. GoalOS separates user objective, mission design, proof production, evidence validation, memory admission, and public commitment.*

6 Proof Mission and Mission Contract

A Proof Mission is the conversion of an objective into an evidence-generating run. Its contract is the first authority-bearing object.

Listing 1: *Mission Contract skeleton.*

```
{
  "mission_id": "mission-001",
  "objective": "...",
  "decision_to_support": "...",
  "success_criteria": [],
  "failure_criteria": [],
  "constraints": [],
  "proof_level": "L3 External Review",
  "risk_class": "medium",
  "allowed_tools": [],
  "required_validators": [],
  "blocked_claims": [],
  "rollback_obligations": []
}
```

The mission contract prevents an agent from optimizing an ambiguous request. It states what the work is for, what would count as success, what would fail the mission, and which claims are blocked before proof.

7 Proof Debt

Proof Debt is the difference between what the system wants to claim and what it can support.

$$D_{proof} = \sum_i \text{Impact}(c_i) \cdot \text{Unsupported}(c_i) \cdot \text{Irreversibility}(c_i) \cdot \text{ReuseRisk}(c_i). \tag{4}$$

A claim may be classified as supported, needs evidence, needs replay, needs reviewer, needs boundary, blocked, or out of scope. This turns uncertainty into a work queue rather than burying it in prose.

8 Custom AGI Jobs

The AGI Job factory starts empty. Jobs are generated only after the mission and proof level are known.

An AGI Job is a proof contract:

Listing 2: *AGI Job skeleton.*

```
{
  "job_id": "agi-job-001",
  "proof_debt_id": "debt-001",
  "source_claim": "...",
  "worker": "Reviewer Agent or Human Reviewer",
  "validator": "AGI Node Validator + Human Review",
  "sentinel": "Boundary Sentinel",
  "deliverables": [],
  "acceptance_tests": [],
  "blocked_claims": [],
  "proofbundle_return_path": "proofbundles/agi-job-001.json",
  "status": "ready-for-execution"
}
```

9 Evidence Docket

The Evidence Docket is not a file pile. It is the proof room. It ties claims to sources, ProofBundles, validator verdicts, risk, replay, and decision state.

Minimum docket contents

Mission Contract; claims matrix; Proof Debt Register; ProofBundle manifest; source provenance; replay manifest; risk ledger; blocked claims; validator notes; Chronicle recommendation; rollback conditions.

A docket is complete only if a reviewer can answer: What was attempted? What evidence exists? What failed? What remains blocked? What may be reused? What must not be claimed?

10 Chronicle Gate

Chronicle is the memory firewall. It admits only candidates that satisfy the selected proof level.

$$\mathrm{Admit}(c,\ell) = \mathbf{1}\{E(c) \geq \theta_E^\ell \wedge R(c) \leq \theta_R^\ell \wedge V(c) \geq \theta_V^\ell \wedge P(c) = 1 \wedge B(c) = 1\}, \tag{5}$$

where E measures evidence strength, R risk, V validation coverage, P replay readiness, and B boundary compliance.

11 Validated Skill Graph

A validated skill is a governed capability object. It is scoped, evidenced, replayable, versioned, and rollbackable.

$$s = \langle id, type, scope, method, evidence, tests, validators, risk, policy, version, rollback, utility \rangle. \tag{6}$$

The graph $G_t = (V_t, E_t)$ contains skill nodes and policy-scoped edges. Edge types include reuse, compose, teach, package, audit, supersede, retire, and future-prior. A future mission may use a skill only if the mission context matches the skill's scope and proof level.

12 Proof Levels

Proof levels calibrate authority. A high-risk public claim, a mainnet handoff, and an exploratory draft should not share one proof threshold.

Level	Name	Required posture	Memory effect
L1	Structural / Demo	Mission structure, explicit Proof Debt, local artifact coherence.	No durable Chronicle influence.
L2	Internal Review	Operator or internal reviewer confirms usefulness, scope, and obvious safety.	Temporary reuse, clearly labeled.
L3	External Review	Cross-source support or independent reviewer / AGI Node validation.	Scoped reusable skill.
L4	Mainnet Handoff	Handoff packet with validated evidence, job specification, and settlement boundary.	May inform external execution preparation.
L5	Chronicle / Formal	Multi-validator evidence, replay, baseline comparison, risk gate, rollback condition.	Durable future mission influence.

Table 1: *Proof levels. Higher levels increase validator burden, evidence strictness, replay duty, and authority.*

The selected proof level is a mission parameter, not a label added after the fact. It controls job decomposition, validator density, Evidence Docket strictness, and Chronicle threshold.

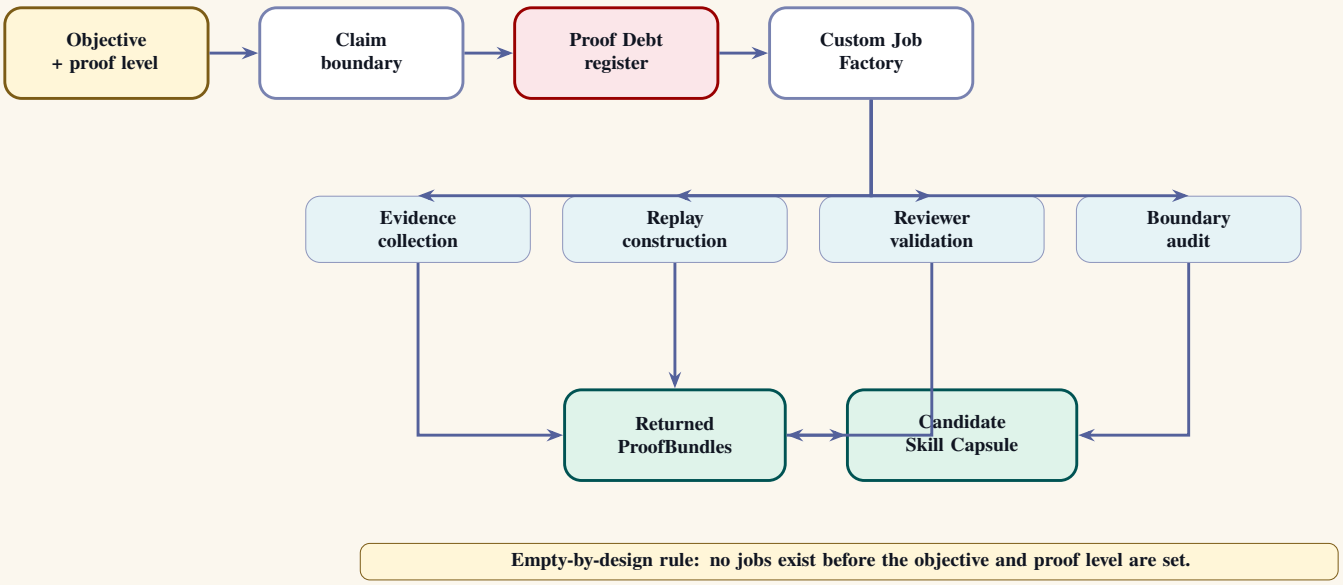

Figure 3: *Proof Debt becomes custom AGI Jobs. The workbench is empty until the mission defines claims, proof level, and boundary.*

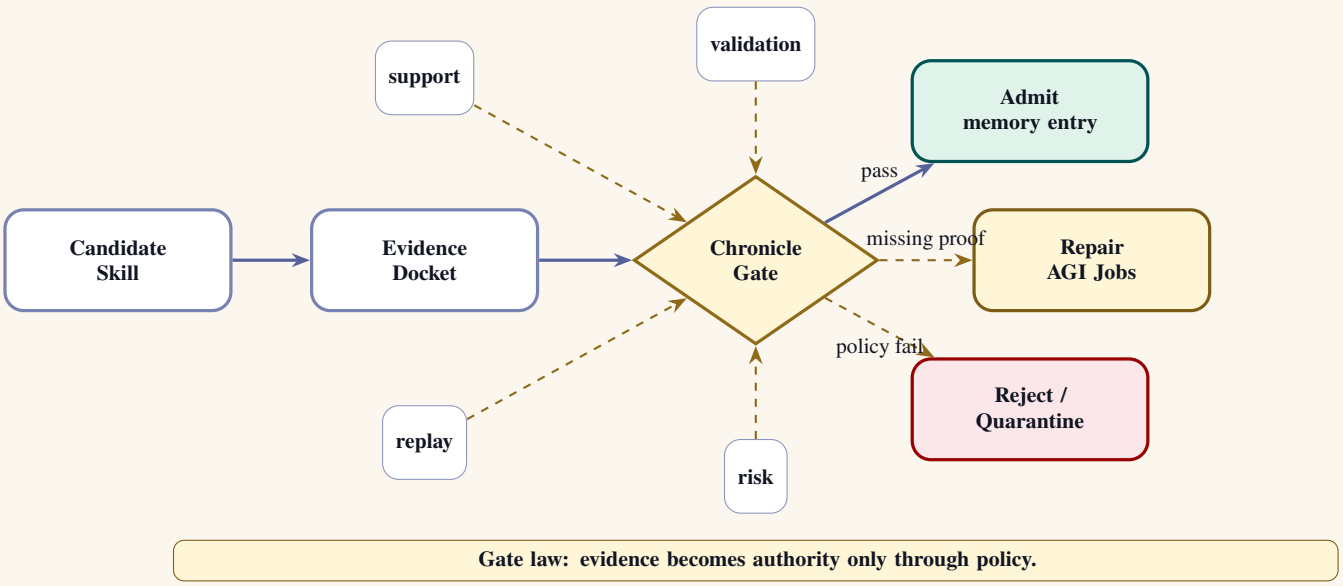

Figure 4: *Chronicle admission. A candidate skill can be admitted, repaired, rejected, or quarantined; unsupported candidates do not become future mission memory.*

13 On-chain Graph Root and Privacy Boundary

The on-chain graph root is a public commitment, not the skill itself. The skill lives in the encrypted capsule, Evidence Docket, ProofBundles, replay manifests, validator attestations, risk ledgers, and Chronicle decision.

A public root record may contain:

Listing 3: *On-chain graph-root record.*

```
{
  "agent": "john.agent.agi.eth",
  "skill_id": "skill-001",
  "graph_root": "0x...",
  "evidence_root": "0x...",
  "proofbundle_root": "0x...",
  "policy_hash": "0x...",
  "schema_hash": "0x...",
  "attestation_root": "0x...",
  "zk_proof_hash": "0x...",
  "version": 1,
  "status": "registered"
}
```

14 Future Mission Improvement

The loop compounds only if future missions learn from admitted skills rather than from raw outputs. A future mission may use a validated skill as a capability prior if and only if:

$$\text{Influence}(s, M') = \text{Admitted}(s) \wedge \text{ScopeMatch}(s, M') \wedge \text{Fresh}(s) \wedge \neg\, \text{Blocked}(s, M'). \tag{7}$$

If any condition fails, the system creates repair work instead of memory influence.

15 Threat Model

The central risks are authority errors, not just software defects.

Threat	Failure mode	Control
Prompt contamination	Persuasive text enters memory as capability.	Chronicle gate; no durable memory without evidence.
Proof leakage	Local draft is treated as validated proof.	Proof level labels, Evidence Docket state, blocked claims.
Replay failure	A result cannot be reconstructed.	Replay manifests, artifact hashes, baseline records.
Validator capture	Weak skill admitted through bad review.	Multi-validator policies, challenge windows, reputation.
Scope drift	Skill used outside admitted envelope.	Scope tags, policy checks, rollback.
Plaintext leakage	Private method or evidence exposed on-chain.	Encryption, commitments, ZK proofs, public/private boundary.
Mainnet confusion	Local handoff treated as settlement.	Explicit wallet, escrow, posting, and settlement boundary.

Table 2: *Proof-loop threat model. The principal failure is unsupported authority over future work.*

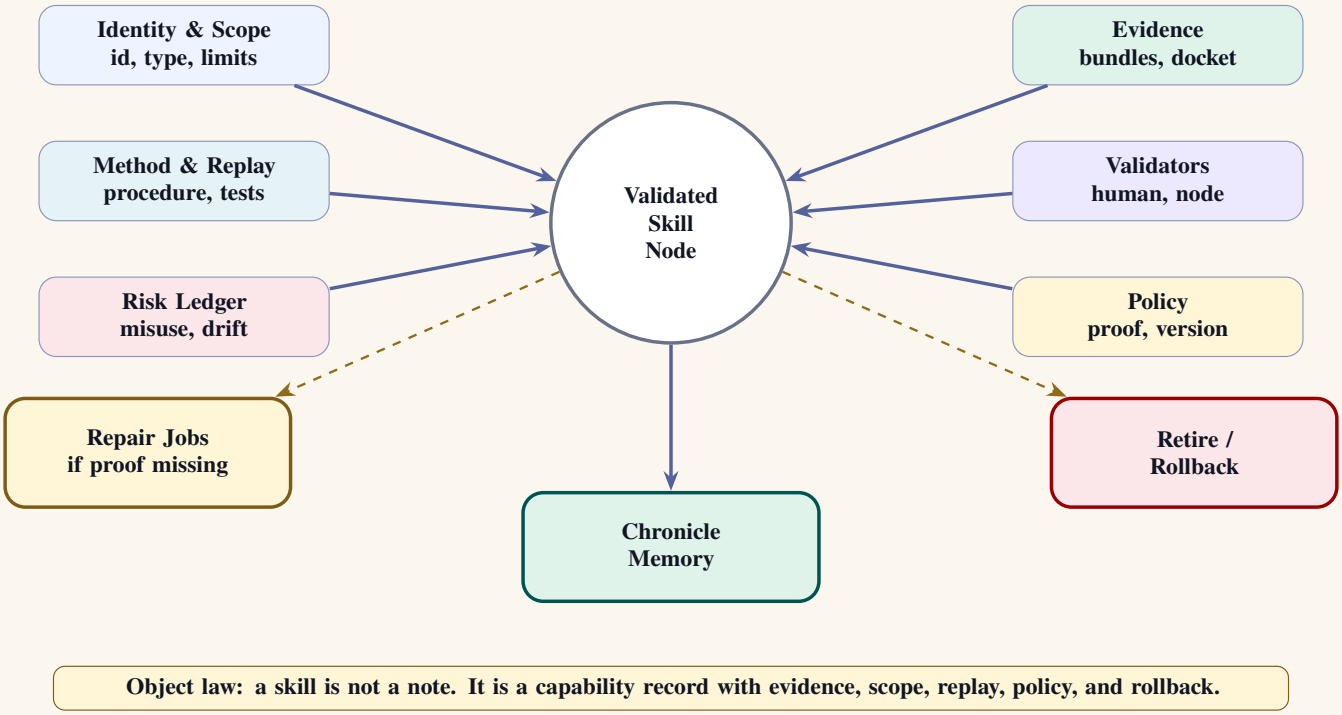

Figure 5: *Validated skill object model. The skill carries operational semantics, proof, policy, and rollback, not merely text.*

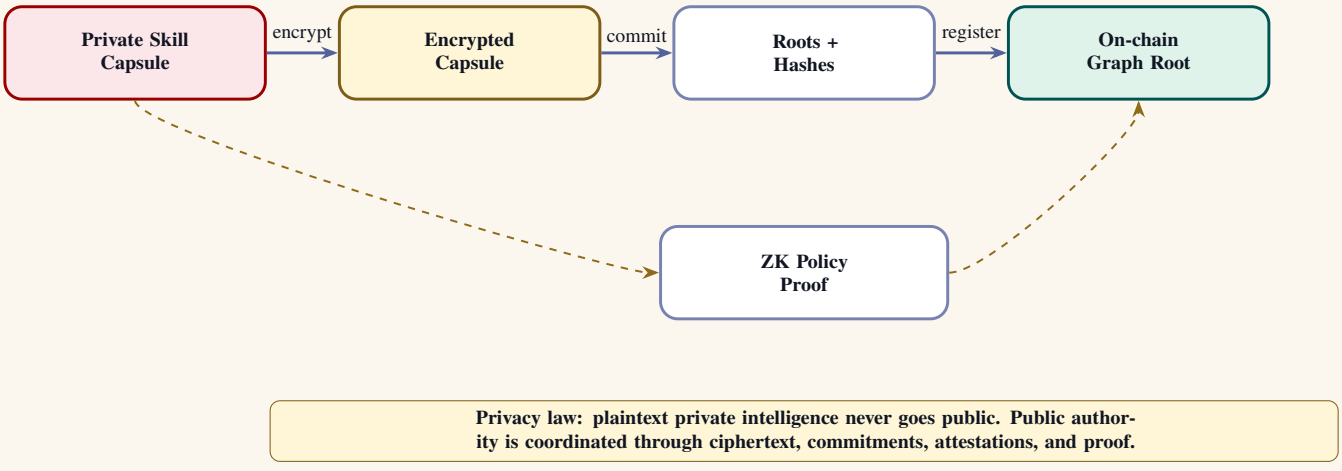

Figure 6: *Public-private proof boundary. Ethereum should coordinate proof-bearing commitments and authorization, not expose private skill intelligence.*

16 Evaluation Program

The system should be evaluated by its effect on future work, not by the number of pages, prompts, jobs, or graph nodes created.

16.1 Metrics

- **Chronicle precision:** fraction of admitted skills that later pass replay or reuse tests.
- **Reuse yield:** improvement in future mission speed, quality, cost, or risk posture.
- **Repair efficiency:** proportion of failed candidates converted into useful AGI Jobs.
- **Drift resistance:** rate at which reused skills remain within scope.
- **Boundary correctness:** absence of local-live confusion or unsupported settlement claims.
- **Evidence density:** proportion of major claims linked to evidence, replay, or explicit uncertainty.

16.2 Falsification tests

The proof-loop claim should fail if any of the following hold under fair evaluation:

1. Admitted skills do not improve future missions compared with notes or prompt libraries.
2. Chronicle admission cannot prevent unsupported claims from entering future mission influence.
3. Custom AGI Jobs do not retire Proof Debt more effectively than a static checklist.
4. The on-chain root boundary cannot be explained to non-technical operators.
5. The system increases confidence faster than it increases evidence.

17 Artifact Standard

A complete run should emit at minimum:

Minimum export

Mission Contract; Proof Debt Register; AGI Job Queue; ProofBundle Manifest; Evidence Docket; Verifier Report; Chronicle Gate Decision; Validated Skill Capsule; Validated Skill Graph Update; On-chain Graph Root Record; Reviewer Room; Action Graph; Claim Boundary; Rollback Plan.

High-rigor runs add baseline comparison, replay manifest, stress tests, validator attestations, ZK policy proof, attestation root, cost ledger, risk ledger, external reviewer notes, and delayed-outcome monitor.

18 Claim Boundary

This paper defines an operating pattern and architecture. It does not claim achieved AGI, achieved ASI, empirical state of the art, legal certification, financial advice, security certification, live wallet support, live settlement, autonomous production authority, or external validation.

Its claim is narrower and stronger: proof-gated loops make autonomous work inspectable, replayable, and governed. Stronger claims require Evidence Dockets, baselines, ProofBundles, replay logs, cost/risk ledgers, validator reports, delayed outcomes, and independent reproduction.

19 Conclusion

`PROOF_LOOPS.md` is the compact law of GoalOS-style autonomous work. It says that useful AI output is not enough. Work must become proof; proof must become evidence; evidence must pass Chronicle; Chronicle must admit a bounded skill; and only then can memory improve future missions.

Final line

Do not remember what was merely said. Remember what was proven, bounded, replayable, and safe to reuse.

Proof today. Capability forever.

REFERENCES

[1] Vincent Boucher. *PROOF_LOOPS.md: Field Notes on Machines That Turn Work Into Capability*. GoalOS operating note, 2026.

[2] Vincent Boucher. *GoalOS Mission OS: The Proof OS for Autonomous AI Work*. Publication-safe, claim-bounded edition, 2026.

[3] Vincent Boucher. *GoalOS: The Validated Skill Graph*. Montreal.AI Research, 2026.

[4] Vincent Boucher. *GoalOS V97.4: Private On-Chain Validated Skill Graphs*. Montreal.AI Research, 2026.

[5] Vincent Boucher. *GoalOS: The Proof-of-Evolution Constitution, AEP-001*. Institutional Standard, 2026.

[6] Vincent Boucher. *AGI ALPHA: A Scalable Substrate for Intelligence Organizations*. Montreal.AI / QUEBEC.AI ecosystem context, 2026.

[7] Vincent Boucher. *GoalOS V85: A Unified Sovereign Proof Economy Operating System*. GoalOS Research Draft, 2026.