Greenseer: Verifiable Event Resolution and Incentivized Rubric Development

Greenseer

September 2026

Abstract

The usefulness of an event contract depends on the reliability of its resolution. Greenseer separates this problem into three components: verifying evidence, searching for proofs, and updating verification rules as new evidence and proof techniques emerge. By combining reproducible computation, adversarial simulation and economic incentives, the system is designed to make resolution risk inspectable before capital is committed. The objective is infrastructure through which agents and humans can fund the discovery of verifiable answers, with explicit safeguards against miner manipulation that would undermine their economic value. The same mechanism enables decentralized fact checking, where collective demand funds the work needed to prove or disprove a public assertion. Together, these capabilities provide a foundation for markets in verifiable knowledge and a continuously improving model of the world. Progress toward superintelligence brings the mechanization of these processes within reach, enabling increasingly autonomous proof discovery, verification and refinement.

1. Introduction

Event contracts require a resolution process connecting external evidence to an enforceable answer. Greenseer develops executable resolution rules, called rubrics, for agents and institutions making economically important decisions. Miners supply programs and evidence; validators generate alternative worlds, reproduce decisions and price surviving attacks before admitting a version at a stated exposure. Simulation exposes ambiguity before capital is committed. Bonds, challenges and versioned revisions create incentives to correct the system's world model as conditions change, supporting increasingly capable, ultimately superintelligent agents.

Agents and humans can simulate resolution before committing capital, examining attacks ruled out under explicit assumptions and the remaining economic risk. The prototype supports zkTLS-style evidence and zkVM-style receipts through notarization and verifier interfaces, currently backed by a trusted proxy notary and development prover signatures. Its risk model accepts externally calibrated detection-probability estimates to price manipulation risk; calibration is an input to admission, rather than a result established by the prototype. Production cryptographic proofs and permissionless improvement rewards remain under development.

A fact-check button on a social post could pool funding to verify an influencer's claim of earning millions trading meme coins. Sufficient demand makes it worthwhile for miners to establish or refute the claim under a stated accounting rule. New evidence or proof techniques can improve verification for subsequent questions.

Signatures identify attestors and execution proofs verify computation; neither establishes the adequacy of the rule or truthfulness of its sources. Verification therefore needs its own process of improvement. The name recalls the greenseers of *Game of Thrones*, who perceive past events without being physically present.

2. Questions, Evidence and Agent Interfaces

Participants may ask questions or propose question families and their rubrics. HTTP and MCP expose requests, quotes, confirmation, simulation, status and receipts; web, terminal and Telegram adapters use the same operations. Wallet delegation bounds spending by amount, budget and expiry. Agents rehearse a question before confirming payment, then replay its receipt.

A claim fixes its statement, parameters, time boundary and authorized signers. Its template fixes the resolution program and evidence policy. Miners collect permitted inputs: notarized web or JSON-RPC responses; chain logs and state observations; EIP-712 or domain-bound EIP-191 attestations; signed institutional reports; and, in experimental paths, attestations over model output or aggregate openings of private commitments. Each route retains its source and trust assumptions.

The target submission binds a frozen program, canonical bundle and simulation results; joint bonded submission is unfinished. Search returns existing verified answers with receipts and identifies unmatched question details.

3. Rubric Construction and Economic Admission

A rubric is a program over explicitly permitted observations. For a canonical claim C, evidence bundle B and pinned version v, its result is

$$R_v(C, B) \in \{\text{TRUE}, \text{FALSE}, \text{REFUSE}\}. \tag{1}$$

REFUSE names missing, contradictory or out-of-domain evidence; unresolved contract settlement uses UNDETERMINED. A simulated *world* assigns facts with truth withheld from the program. An *admission* binds a tested version to economic conditions. [1, 2]

3.1. Definition, World Generation and Miner Feedback

The validator defines the predicate, probes the real source and enumerates boundaries: timestamps, units, identities, finality, completeness and alternative explanations. It supplies the world interface, truth function, ordinary faults, adversarial modifications and attack-cost assumptions. Simulated and real worlds expose the same permitted methods. Adversarial changes alter what the source reports while leaving the hidden truth unchanged.

The miner declares permitted reads, verdict semantics and a candidate program, then addresses concrete failures. The validator runs visible cases, configured digest-seeded hidden cases and historical bundles, returning honest-world errors, successful attacks, undeclared reads and excessive refusals. Revisions narrow the predicate or add authenticated observations: a transfer establishes a flow, for example, without establishing a sale.

3.2. Record, Replay and Simulation

The recorder captures permitted observations in a canonical bundle. A separate interpreter checks the program digest and replays the decision from those bytes. Divergent replay or undeclared reads require revision. This isolation supports reproducibility; hostile-code security remains unfinished.

Perturbing individual facts, removing leaves and testing boundaries reveals which observations determine the answer. Crashes fail evaluation. Seeded worlds, preserved bundles, frozen versions and repeated evaluations control execution and review variability, without eliminating factual uncertainty. The optional model judge pins weights, engine, sampling and grammar; computed facts reduce reliance on interpretation.

3.3. Pricing Residual Attacks

The customer supplies a *weight*: value per answer, reuse of the source, relevant adversary and tolerable refusal rate. For every observed surviving attack k, admission requires

$$c_k + q_k K \;\geq\; 3(1 - q_k) V a u_k. \tag{2}$$

Here c_k is effort plus the cost of keys the adversary does not already hold; q_k is the assumed detection probability; K is the loss on detection; V is exposed value per answer; a is answers per source unit; and u_k is units compromised. The factor three is the laboratory safety margin. All estimates are disclosed assumptions. The smallest residual-implied value ceiling binds; repeated use of one forged source increases the exposure. When no attack succeeds, admission remains bounded by the attack budget tested.

A wrong definite answer on an honest world, a replay violation or excessive prior-weighted refusal produces REVISE. A residual below the safety hurdle produces REJECT; satisfying the checks produces ADMIT for that weight. The exit report additionally displays all adversary classes, including the poster, the binding ceiling, cheap residuals, untested attack rows and old-version regressions. These reports make omissions visible; passing finite tests does not establish universal correctness. The governing principles are explicit semantics, reproducible evidence, accountable assumptions, useful coverage and measured simplicity.

4. Settlement and Incentives

A claim pins its statement, rubric, judge and execution image. The miner commits before revealing a payout-bound receipt; verification checks claim fields, evidence and signatures. A verified counter-proof for the same claim and program must give the opposite definite verdict, or a definite answer against UNDETERMINED. If the arbiter upholds the challenge, the challenger becomes the miner, receives the wrong proposer's bond and opens a fresh challenge window. A failed challenge pays its bond to the upheld miner, or burns it for UNDETERMINED. Disputes permit two rounds; seven days without a ruling ends the claim as UNDETERMINED. [3]

A definite-answer miner earns the bounty plus 4% of the losing pool; another 1% funds the reserve or lineage author. Lineage claims additionally pay a law fee of 5% of bounty, minimum one token unit. UNDETERMINED refunds stakes and bounty, pays no answer reward and burns the law fee. Bettors and the poster cannot mine that claim. Payout binding and commit–reveal limit copying.

The Solidity registry requires an active author bond of at least 1,000 units. Challenges require 50 units or 5% of active bond, whichever is greater. An upheld lineage challenge freezes the lineage and slashes

$$S = \min\big(B_{\text{total}}, \max(200, 4D)\big), \tag{3}$$

where B_{total} is active bond plus queued withdrawals and D counts losing pools plus bounties decided over 30 days, including the challenged claim even if older. The finder receives $\min(S/2, \max(100, S/10))$; the remainder burns. A bonded arbiter earns direction-independent ruling fees and loses collateral for missed deadlines. Exposure caps, delayed withdrawals and evidence-availability challenges constrain liabilities. These contract balances are distinct from the laboratory's simulated admission bonds.

5. Learning and Iteration

5.1. Versioned Improvements

Each claim pins a rubric version; lineage revisions improve future decisions without changing existing claims. Ruled failures become regression fixtures; new observation routes create evidence leaves and versions. Prior bundles remain available for comparison. [2–4]

Review starts from the current rule, investigates outside it, states a numeric prior and changes one named object: definition, ruled world, version, residual cost, guarded assumption, evidence leaf or contextual world-model row. Context alone cannot change a verdict or earn payment. Reviewers rerun affected and out-of-sample cases and measure code, inputs, refusal sites and tuning parameters.

Laboratory admissions accept challenges drawn from their own batteries: a successful challenge revokes admission and records a slash; failure forfeits the challenger's simulated bond. Solidity lineage fees reward authors, but patches remain author-controlled. Permissionless improvement royalties and bonded improvement submissions are unfinished.

5.2. Lessons and Remaining Incentive Work

Repeatable but incorrect model answers motivated computed facts and stricter evidence policies. Omitted review comparisons motivated executable exit reports. The living PnL fork preserves its baseline for changed-decision comparison.

The earlier project paper reports three adversarial learning rounds; the third paid a malicious improvement while an honest proposal lost its bond. The reported weaknesses were incomplete changed-decision coverage, digest-dependent counterexamples and self-reported costs. Proposed repairs include canonical arbitrary-world challenges, changed-decision checks and independent cost experiments. Proposed upkeep pays admitted improvements and future author fees; paid standing reviews renew or lapse admission, with lapsed rules refusing new answers. Their incentive compatibility remains unestablished.

6. Applications and Current Implementation

Nineteen registered laboratories share recording, replay and admission machinery. Registration does not imply production readiness or admission at every exposure. [1, 2]

Application	Implemented laboratory families
Trading operations	Private and public Hyperliquid PnL; PnL reconciliation and its living fork; position reconciliation; hedge-break detection; missing PnL legs.
Market events	Upbit and generic market listings; token supply changes; Chainlink closes; feed staleness; attributed treasury flows; contact with named hacker-wallet sets.
Attestations	Deployed software versions; aggregate attested receipts; firepit searcher PnL; firepit activity and cartel-related attestations.
Reference lab	Toy ledger for exercising the shared rubric-development path.

Table 1: Laboratory scope in the September 2026 snapshot.

Sandbox templates additionally exercise delivery, registry and DNS facts, notices, posts, signed metrics, JSON fields, reconciliation, transfers and hosted-model responses. Kernel laws cover transaction receipts, balances, transfers, event counts, contract code and PnL breaks.

Scope remains literal: public trader PnL requires a stated accounting convention and master-key consent; it does not certify Sharpe. Supply changes can refuse unsupported rebases. Treasury flow does not establish lending intent; activity patterns and attestations do not establish collusion. Reconciliation may refuse incomplete account coverage.

6.1. What Has Been Built

The prototype implements the interfaces, recording/replay, adversarial worlds, perturbation, pricing, exit reports, receipt search and hash-chained journal described above. Solidity supplies settlement and lineage contracts; local-chain rehearsals, tests, real-source readers and preserved evidence accompany the source. These are implementation observations, not newly certified test results.

The development verifier accepts a prover signature, not a STARK or zkVM execution proof. Web evidence uses a trusted proxy notary, not production TLS transcript proofs. The prototype registry retains a designated verifier and arbiter. Recording completeness, hostile-code isolation and failure handling need further work. No public production deployment is evidenced.

6.2. Event Contracts and Future Work

An event contract about the future can select its resolution rubric before funding and invoke it after the event. The same simulation environment can expose ambiguous boundaries and failure conditions in advance. Prediction markets therefore extend the resolution primitive, while requiring additional future-event timing and market-lifecycle support beyond the current past-event posting flow.

The immediate development priorities are production evidence and execution proofs, independently tested attack costs, canonical challenges and adversarial validation of improvement rewards. Greenseer's contribution is a common process for turning a question into a reproducible decision, stating the economic boundary of that decision and preserving what is learned when it fails.

Source notes. [1] Supplied project *README* and site documentation. [2] *Rubric Labs README*, core evaluator, bundle/replay, perturbation, exitcheck and registry source. [3] *ClaimMarket.sol* and *RubricRegistry.sol*. [4] Existing project whitepaper and living-PnL source. The linked *PROCESS.md* and incentive scorecards were absent. The companion research reference maps the available sources and reconstructed process.