
Keeping the Judge Off the Edit Surface: Gated Recursive Self-Improvement on a Strong-Evaluator Domain

Thomas Wang

thomas@openthomas.com

github.com/PredictionMarketTrader/openthomas

Abstract

Near-term recursive self-improvement (RSI) is optimization of an agent’s *harness*, and its first bottleneck is evaluator strength. But a strong evaluator only relocates the danger: an optimizer that can edit its own judge will lower the bar rather than clear it. We present the self-improvement design of **OpenThomas**, an open-source trading agent for weather prediction markets — a domain with an unusually strong evaluator (daily, objective NWS settlement; leak-free replay). Its codebase is split into a *kernel* plane (promotion gate, parameter bounds, risk rails, truth pipeline) that only the operator writes, and an *agent* plane (decision parameters, forecast prompt, strategy code) that an evolution loop writes *only through the kernel gate*. The split is the security model: the judge is structurally off the optimizer’s edit surface. The gate scores candidates on leak-free replay with four anti-gaming properties (held-in/held-out splits, totals-not-ratios scoring, out-of-sample regret rollback, and a calibration veto), and a single accept mechanism absorbs a growing set of mutation operators. In a leak-free replay over 968 settled markets, the disciplined production rule turns a naked model-vs-market loss into a positive expectancy by *selectivity* — the same evaluator that gates self-improvement also quantifies where the edge is.

1 Introduction

Framing near-term recursive self-improvement (RSI) as optimization of the *harness* — how an agent plans, acts, remembers, and is evaluated — identifies *evaluator strength* as the first bottleneck [1]. Most RSI testbeds lack a strong evaluator, so “improvement” is measured by a proxy the optimizer can drift. We start from a domain that supplies an unusually strong one, and confront the failure mode that a strong evaluator merely relocates: not a weak judge, but a *gameable* one. An optimizer that can edit its own judge finds it easier to lower the bar than to clear it.

Our testbed is **OpenThomas**, an open-source autonomous agent that trades daily station-temperature markets on Kalshi and Polymarket. These markets settle every day against an objective public record (the U.S. National Weather Service climatological report for one named station), which makes the evaluator *dense*, *objective*, and — with historical as-of forecasts and order books — *reproducible in leak-free replay*. The agent embodies a “harness is the edge” thesis: every frontier language model given real capital to trade prediction markets has lost money [2], so OpenThomas confines the language model to a bounded adjustment and puts selectivity, calibration, and risk in deterministic code. Here we describe the part that makes it a self-improving system, and the structural guarantee that keeps self-improvement honest.

Contributions. (i) A self-improvement architecture whose safety is *architectural*: a kernel/agent plane split keeps the evaluator off the optimizer’s edit surface, enforced today by code structure and

Plane	Contents	Written by
kernel	promotion gate, parameter bounds, risk engine & safety rails (sizing, exposure caps, kill-switch), truth pipeline, journal schema	operator only
agent	decision-rule parameters, forecast prompt template; later: workflow structure, strategy code, the evolver itself	evolution loop, <i>via the gate</i>

Table 1: The plane split is the security model. One line is drawn precisely: entry-*selectivity* knobs (minimum edge, market-prior blend weight) are genome parameters under kernel bounds, while sizing, exposure caps, and the kill-switch are safety rails that can never enter the parameter space — a kernel policy test fails any change that tries to move one there.

on a roadmap to OS-level process isolation. (ii) A promotion gate with four anti-gaming properties on leak-free replay, and a single accept mechanism that absorbs a growing set of mutation operators (decision parameters and forecast prompts shipped; workflow, code, and weights on the same gate). (iii) Evidence from the same evaluator: a leak-free replay ablation in which discipline, not a sharper model, produces the edge.

2 Weather markets as a strong-evaluator domain

Three properties are exactly what RSI asks for and rarely gets. **Dense, objective settlement:** each market resolves against one station’s observed daily extreme, once per day, with no interpretive latitude; a desk on ten U.S. cities sees dozens of ground-truth points per day rather than a few per year. **A learnable local edge:** markets settle on a *specific* station, and global forecast models carry systematic, station-specific biases there (a 90-day hindcast found Miami consensus 1.8–2.1 °F cold, Philadelphia +2.3, LAX −1) that are estimable from free public data, offline. **Leak-free reproducibility:** historical as-of guidance and historical order books let a replay present the agent exactly the information it would have had, and no more — so a candidate rule, and the whole self-improvement loop, can be scored without the replay silently cheating.

3 The structural guarantee: kernel and agent planes

Self-improvement is safe only if improvement cannot redefine “improvement.” So the codebase is split into two planes, and the split *is* the security model (Table 1). The **kernel** plane — the promotion gate, parameter bounds, the risk engine and all safety rails, the truth pipeline, and the journal schema — is written only by the operator; the evolution loop has no write path to it. The **agent** plane — decision-rule parameters, the forecast prompt template, and (on the roadmap) workflow structure, strategy code, and the evolver itself — is written by the evolution loop, but only *through* the kernel gate.

The operator is moved *up* the stack: they set bounds and review lineage, and are forbidden from hand-tuning trading parameters — that is what the evolution loop is for. Today the split is enforced by code structure (the LLM proposer only ever returns JSON; every file write happens in deterministic code after the gate has ruled). The deployment roadmap makes it structural in the OS: kernel paths owned by the operator’s user, the evolver running as a user with write access only to the agent plane, gate decisions crossing a process boundary, and every promotion an auditable commit authored by the loop itself.

4 The evolution loop and its gate

Three loops nest (Figure 1). A fast *trading* loop runs the champion parameters and never waits on the others. A *reflection* loop distills settled trades into bounded playbook rules. The *evolution* loop mines failure evidence, proposes mutations — LLM-directed by the evidence, plus Gaussian mutations from an *archive* of past variants (Darwin Gödel Machine-style [3], against diversity collapse) — and submits them to the gate.

The gate (kernel, frozen) scores a candidate *strategy callable* on leak-free replay with four anti-gaming properties. *Held-in/held-out:* winners are picked on older days; the freshest settled days

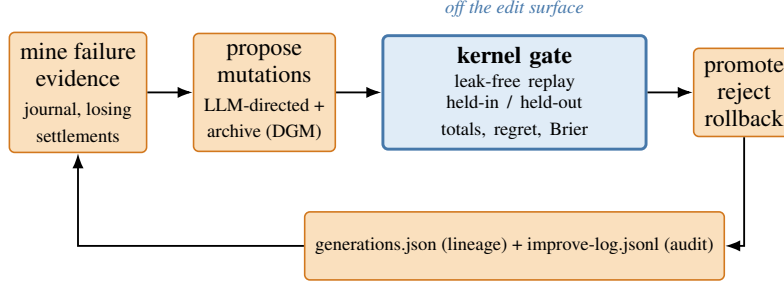


Figure 1: The evolution loop: propose–evaluate–accept with a Darwin Gödel Machine-style archive of past variants as mutation parents. The gate (kernel, shaded) is the only path to the agent plane and is never on the optimizer’s edit surface. Rejected and rolled-back generations stay on file as future parents.

Decision rule (leak-free replay, fees in)	Trades	Win	PnL/ct
Naked model-vs-market	425	41%	−\$0.007
+ learned station bias	396	39%	−\$0.005
+ market-prior blend (production)	184	45%	+\$0.028

Table 2: Replay over 968 settled markets (2026-06-24 to 07-14; timing-safe snapshots; bias learned out-of-window). The market-prior blend more than halves activity and flips expectancy positive: *selectivity* is the edge. Small sample, paper mode — an existence proof, not a track record.

only get a veto, and the window rolls forward daily, so repeated meta-cycles face new data rather than re-fitting one static split. *Totals-not-ratios*: scoring is total replay PnL with a minimum trade count — a rule that trades twice cannot win on ratio, a rule that trades nothing cannot win on drawdown. *Out-of-sample regret rollback*: each cycle first re-scores the active generation against its parent on the fresh window and reverts if the parent now wins — an overfit promotion is undone before anything new is proposed. *Calibration veto*: for the forecast operator, a prompt that wins PnL while worsening Brier got lucky, and luck does not promote. Generation 0 is the operator’s config and applies no overrides.

One gate, growing write surface. Improvement axes are unified as mutation operators feeding the *same* accept mechanism: playbook rules and calibration refits (shipped), decision-parameter and forecast-prompt mutation (shipped), then workflow structure, then agent-plane code diffs (which add shadow paper-trading and human diff review), then LoRA/GRPO weight updates with a time-discounted reward $R \cdot e^{-\lambda \text{ days-to-close}}$ — a trained adapter is just another candidate at the same gate. The admission rule is strict: a parameter enters the space only when the gate can actually discriminate on it. *Tuning what you cannot measure is drift, not improvement*. This is why code diffs come last (they need the strongest evaluation: replay *plus* shadow trading *plus* human review), and why the enterprise is anchored to a strong-evaluator domain in the first place.

5 Evidence from the same evaluator

The gate’s trustworthiness rests on one discipline: *never let the replayed market know something the replayed model does not*. Station bias/ σ are learned strictly before the replay window; prices come from snapshot candlesticks that precede the temperature extreme being bet on (a single timing leak once turned a losing rule into a fake winner; fixing it turned it back). The same leak-free pipeline that scores candidate generations also lets us quantify where the edge is (Table 2): over 968 settled markets, a naked model-vs-market rule barely loses, but blending the model with the market prior more than halves activity (425 $\rightarrow$ 184 trades) and flips expectancy positive. Selectivity, not a sharper model, is the edge — the harness thesis, measured by the same judge that governs self-improvement.

6 Failure modes and limitations

Each classic RSI failure mode has a structural counter here. *Reward hacking* → the judge (gate, truth pipeline, risk engine) is off the edit surface, with hard bounds and totals-not-ratios scoring. *Overfitting the evaluator* → rolling held-out veto plus regret rollback on post-promotion data. *Diversity collapse* → archive-parent random mutations alongside LLM-directed ones. *Weak evaluator* → the domain choice itself; operators without replay data get “insufficient data,” not a silent promotion. *Runaway autonomy* → the drawdown kill-switch, paper-by-default, and live-mode two-step opt-in are all kernel and cannot be loosened by the loop; a dead LLM endpoint degrades evolution to random mutations, and a dead evolution loop degrades OpenThomas to a static (still disciplined) trader.

Limitations are real. Replay omits the news brief, intraday observations, and the calibration layer; it measures the template’s skill at adjusting the statistical baseline, the dominant live pathway for weather markets, not the whole live agent. The results are small-sample and paper-mode — an existence proof about where the edge comes from, not a track record; the reference agent’s live run is the ongoing public test. And the strong-evaluator property that makes this domain tractable is what most other markets lack; extending the gate to weaker-evaluator settings (via archived-input trace replay) is future work. The system and its live feed are public.

References

- [1] L. Weng. Harness Engineering for Self-Improvement. Blog post, 2026.
- [2] Prediction Arena: evaluating frontier language models as autonomous prediction-market traders. arXiv:2604.07355, 2026.
- [3] The Darwin Gödel Machine: open-ended self-improvement with a variant archive. 2025.