An LLM-Assisted Genetic Algorithm over Heuristic-Algebra Genomes

A Cross-Market Empirical Study

Henry Carstens
Builder, Agentic Quant, Inventor of Heuristic Algebra
hcarstens@gmail.com hcarstens.github.io

Version 0.1, 2026-04-29

Abstract

This paper reports a cross-market empirical study of LLM-assisted genetic-algorithm search where the search space is encoded as a *Heuristic-Algebra genome*: each candidate strategy is a Python class synthesized by a large language model from a population whose seed pool is drawn from typed compounds in the Heuristic Algebra $\mathbb{H}$ [1].

We evaluate the scaffold across six futures markets (HG copper, CL crude, ES SP500, TY 10-year notes, ZS soybeans, GC gold) under a canonical reproducible protocol with three random seeds and two filter cohorts. Three findings dominate:

1. **Construction-method ordering matches run-stage ordering.** The selection pressure encoded in $\oplus_{\text{bio}}$ (BPD) carries through to run-stage fitness; $\oplus_{\text{ha}}$ (MC) and $\oplus_{\text{inn}}$ (RPT) compounds underperform on the same market, by margins consistent with each operator's compression mechanism.
2. **Encoding scale-up dominates capability scale-up.** Adding pre-computed primitives (`atr14`, calendar features) to `self.data` lifted mean Sharpe on HG from 0.46 to 0.87 on ES. Replacing the LLM brain (Gemini Flash-Lite → Claude Sonnet 4.6) at 33× the per-run cost produced a *19% lower* mean Sharpe across the same five markets — an indicative null result.
3. **The negation operator ¬ is a generative-loop antipattern.** Across 635 candidate children produced via LJ7-style negate-mode, zero exceeded the parent score. The operator is productive in judgment loops (where flipping a belief tests robustness) but destructive in generative loops (where the parent already encodes earned signal).

The scaffold and all run artifacts are public. Reported returns deduct a fixed transaction cost of 0.0008 (8 bps) per side.

Appendix A: Results Gallery (Pre-Introduction)

This appendix is intentionally placed before Section 1 per request. This gallery now shows only ES.

ES

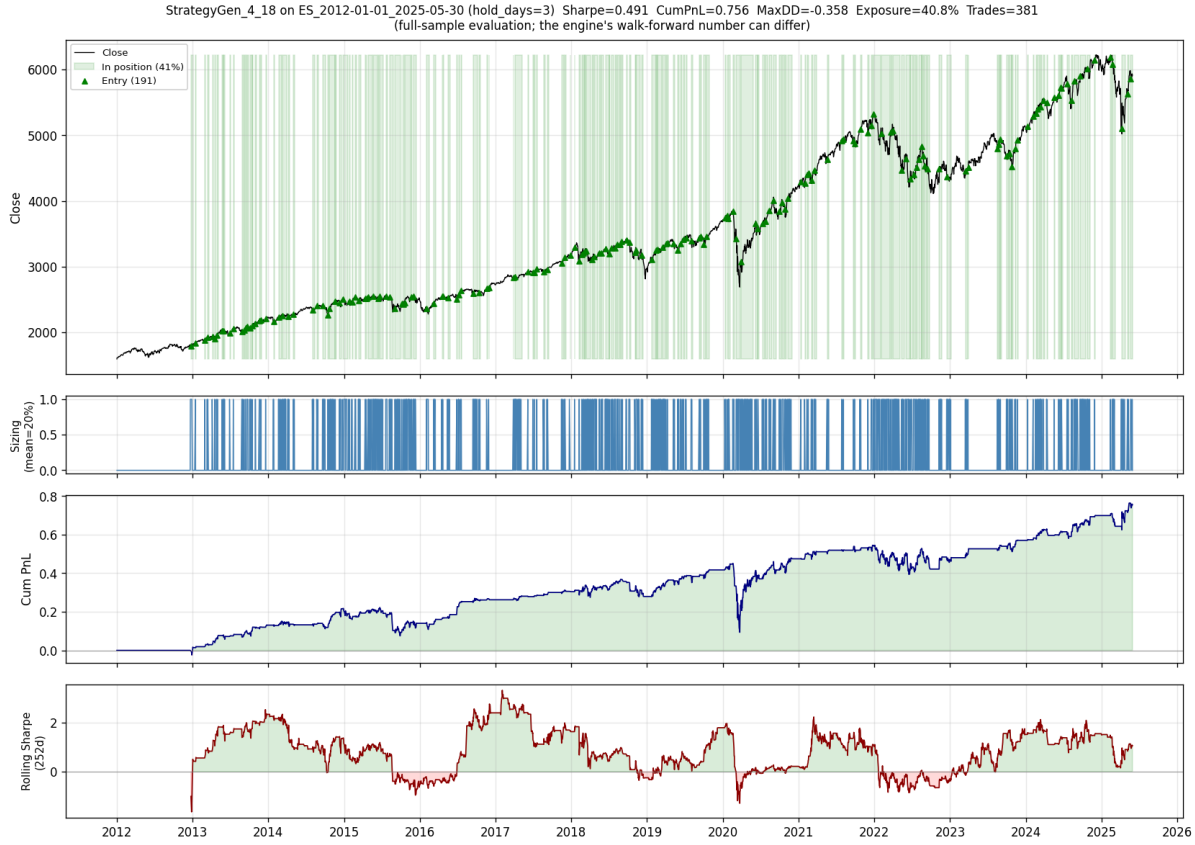

Figure 1: ES: `results/evolution/run_20260428_214033_ES/trades.png`

1 Introduction

The Heuristic Algebra $\mathbb{H}$ [1] is a formal system for manipulating fields of study via their axiom sets. Its operator set $\{\oplus, \oplus_{\text{bio}}, \oplus_{\text{ha}}, \oplus_{\text{inn}}, \neg, \sim, \leftrightarrow, \perp, \pi, \delta, \uparrow\}$ produces typed compounds (lean, harmonic, innovative) from any pair of source heuristics, and supports projection to focused subsets, decompression for context-shifting, and Bayesian update against evidence. A core claim of $\mathbb{H}$ is that combination is *operator-dependent*: the same source axioms yield structurally different outputs under $\oplus_{\text{bio}}$ vs. $\oplus_{\text{ha}}$ vs. $\oplus_{\text{inn}}$, each suited to different downstream purposes.

Genetic algorithms [6, 3] are population-based search procedures over an encoded chromosome space. The $\mathbf{H}_{\text{GeneticAlgorithms}}$ axioms (GA1–GA7) frame a canonical structure: encoding (GA1) *is* the search space, fitness (GA2) is the sole communication channel, crossover (GA4) and mutation (GA5) exploit and explore, and elitism (GA7) plus diversity (GA6) sustain the population. Recent work [4, 5, 8, 7] has shown that LLMs can serve as the *recombinase* for symbolic GA chromosomes, producing semantically valid offspring where classical bit-flip operators fail.

This paper combines the two systems. We treat each candidate trading strategy as a chromosome (a Python class implementing a regime gate), and we draw the seed pool of the population from typed Heuristic-Algebra compounds: Behavioral Price Dynamics (BPD, via $\oplus_{\text{bio}}$), Market Counterpoint (MC, via $\oplus_{\text{ha}}$), Reflexive Price Theory (RPT, via $\oplus_{\text{inn}}$), and others. The LLM acts as crossover and

mutation operator. The walk-forward evaluator on historical futures bars provides fitness. We then run the scaffold under a canonical reproducible protocol across six futures markets spanning four asset classes (metals, energy, equity, bonds, agriculture) and report findings.

The research is a *report on a research scaffold*, not a strategy proposal. We are explicit about what the visual-pass rate, dud filter, and small-N cohorts mean for evidence quality.

Contributions

1. A working LLM-assisted GA scaffold over Heuristic-Algebra genomes, with per-prefix multi-armed bandit, Pareto-front retention, LLM-driven crossover with $A + B$ attribution, and a quality-flag/visual-scan disconfirmation gate.
2. The first cross-market empirical evidence that construction-method ordering ($\oplus_{\text{bio}} \succ \oplus_{\text{ha}} \succ \oplus_{\text{inn}}$) propagates from construction-stage selection pressure to run-stage fitness.
3. A six-market compendium showing that genome widening (one day of work, \$0 marginal cost) produces larger Sharpe lifts than substituting a frontier LLM for the budget LLM (one week of work, 33× per-run cost).
4. Empirical falsification of the LJ7 negation operator within generative loops: 635 candidate children, 0 wins.

2 Heuristic Algebra: The Genome's Source

We give a brief operational summary; full definitions appear in Carstens [1].

2.1 Operators

The relevant operators for this study are:

- $\oplus$ **(Combination):** $\mathbf{H}_{A \oplus B} = \mathbf{H}_A \cup \mathbf{H}_B$. Classical union of axiom sets.
- $\oplus_{\text{bio}}$ **(Biological Selection):** lean compound — cull the unfit, retain conserved genes (axioms appearing $\sim$ across multiple sources), resolve all contradictions ($\perp$). Output: stable equilibrium, fit minimum-viable axiom set.
- $\oplus_{\text{ha}}$ **(Harmonic Arrangement):** rich compound — preserve productive contradictions ($\perp_{\text{productive}}$) as generative engines, compose independent voices, scope dominant axioms via *harmonic sacrifice*. Output: dynamic equilibrium with preserved tensions.
- $\oplus_{\text{inn}}$ **(Innovative Recombination):** novel compound — generate emergent axioms not in either source via creative lenses (cartography, poetry, beauty), then prune to a parsimonious set.
- $\neg$ **(Negation):** replace a single axiom with its logical complement. Deterministic given a sufficiently precise axiom.
- π **(Projection):** select a k-axiom subset under explicit validity criteria (source coverage, negation inclusion, contradiction resolution, functional completeness, precondition transparency).
- $\uparrow$ **(Update):** Bayesian update of axiom credences against evidence; the only operator that lets a compound *learn*.

The construction operators $\oplus_{\text{bio}}$, $\oplus_{\text{ha}}$, $\oplus_{\text{inn}}$ are **non-commutative in output**: the same inputs produce different compounds under each operator. Each operator encodes a different selection pressure on the source axiom sets.

2.2 Compounds Used

We construct three compounds from related axiom families:

Compound	Operator	Source heuristics	Axioms
BPD (Behavioral Price Dynamics)	$\oplus_{\text{bio}}$	Behavioral Economics, Markets, Risk Management	7
MC (Market Counterpoint)	$\oplus_{\text{ha}}$	Markets, Psychology, Decision-Making	8
RPT (Reflexive Price Theory)	$\oplus_{\text{inn}}$	Markets, Behavioral Economics, Complex Systems	7

Each compound was constructed from substantially overlapping source material; the operator choice differentiates the output. BPD's seven axioms (scarcity, equilibrium-as-attractor, cognitive frugality, herding, confirmation bias, probabilistic determinism, scale separation) are each *independently fit* — each maps to a discrete operationalizable predicate on price-and-volume data. MC's axioms preserve a productive tension between rational pricing (MC1) and cognitive frugality (MC4), classifying the deviation as the signal. RPT's axioms (reflexive forecast, bias dynamics, capital-weighted behavioral field, etc.) are dynamical-systems frames that aggregate multiple market participants into emergent state variables.

2.3 Why Construction Method Matters Downstream

The central observation that motivates this paper: **the construction-stage selection pressure of $\oplus_{\text{bio}}$ produces seed pools whose axioms each survive a *second* round of selection pressure** (the GA fitness gate). The construction operator's compression mechanism is itself a structural prior on the GA's diversity properties.

- $\oplus_{\text{bio}}$ produces seven *orthogonal* axioms — each is independently testable on OHLCV data.
- $\oplus_{\text{ha}}$ produces eight *interdependent voices* — each axiom is partially defined by its relationship to the others.
- $\oplus_{\text{inn}}$ produces seven *novel-but-untested* frames — generated by aesthetic/innovation-driven projection rather than fitness pressure.

A breeder that samples one axiom at a time from each pool encounters a qualitatively different search space depending on the pool's construction.

3 LLM-Assisted Genetic Algorithm

3.1 The Scaffold

The scaffold (`src/evolution/`) implements an evolutionary loop with population $N = 30$ and up to 24 generations under adaptive early-stop.

Encoding (GA1). A candidate is a Python class inheriting from `StrategyBase`, overriding the method `decide_size(self) -> pd.Series` which returns a per-bar gate value in $\{0, 1\}$ (binary mode) or $[0, 1]$ (continuous mode). The class operates on `self.data`, a `pandas` DataFrame with columns:

- Base data: `close, high, low, open, volume, returns`
- Pre-computed primitives (*the widened genome*): `atr14`, `atr14_ratio`, `day_of_week`, `month`, `week_of_month`, `days_to_month_end`, `days_to_year_end`

A fixed entry trigger `base_trigger` $\in$ {down_day, neutral} and a fixed hold horizon `hold_days` apply outside the candidate's control. The candidate's `decide_size()` output is multiplied with the trigger and held forward via `rolling(hold_days).max()`, then shifted by 1 for $t{+}1$ execution. The encoding deliberately constrains the candidate to the *gating* decision: which trigger fires get acted on.

Fitness (GA2). Walk-forward evaluation produces six metrics per candidate: Sharpe, CAGR, MaxDD, regime robustness, decorrelation, and exposure penalty. The composite score is a weighted sum (0.45 / 0.30 / 0.20 / 0.15 / -0.30 / -0.20) with two hard penalties: a min-trades floor (200 trades, weight 0.5) and a no-trade clamp on `regime_robustness` and `decorrelation`.

Selection and Elitism (GA3, GA7). A high-water-mark winner is preserved across all generations (strict elitism). A Pareto front of size 5 is also retained, non-dominated over (Sharpe, CAGR, $-|\text{MaxDD}|$) — this is the LJ3 disconfirmation gate.

Crossover (GA4). The LLM produces 2-parent recombinations via a dedicated prompt that shows both parents' code and metrics. Source attribution: `cross:A+B`, distinguished from `breed:parent` (single-parent mutation) and `escape:parent` (negate mode).

Mutation (GA5). The LLM produces single-parent mutations via a critique prompt including the parent's code and per-market metrics.

Diversity (GA6). A multi-armed bandit over heuristic-prefix arms (BPD, MC, RPT, FS, Mk, P, SP, Ha) with ε-greedy selection ($\varepsilon = 0.30$) and `win_rate` reward (1 if score $>$ parent, else 0). A graduation rule excludes arms from the exploit branch after N pulls with 0 wins.

3.2 Encoding Boundaries

The genome explicitly *omits*: entry direction (always long), entry trigger (hardcoded), hold horizon (fixed), exit logic (mechanical hold expiry), and cross-market regime classification. These boundaries are deliberate research-stage simplifications.

3.3 Quality Flags as the Visual-Scan Gate

After each run terminates, a `quality` block is computed:

- `flags`: advisory list — `low_accepts` (accepts < 4), `early_plateau` (last_improvement_gen $<$ 6), `extreme_cagr` ($|\text{CAGR}| > 1.5$), `high_cagr_to_sharpe` ($|\text{CAGR/Sharpe}| > 1.0$), `low_consistency` (rolling-area ratio < 0.10).
- `requires_visual`: true if any flag fires.
- `visual_pass`: user-set after manual review of `trades.png`, `null` otherwise.

The flag-block design is **flag-not-exclude**. The canonical protocol filters on `visual_pass`, not on the auto-flags. Flags are advisory; the visual scan is the gold standard. A run with `visual_pass =`

`null` ∧ `requires_visual = true` is excluded from canonical evidence with a "needs review" note — never silently included or rejected. This is the empirical implementation of LJ3 (asymmetric disconfirmation).

3.4 The Canonical Evaluation Protocol

A full canonical comparison fixes every knob except the comparison variable. Specified in `config/evolution/canoni` `output_mode: binary`, `hold_days: 3`, `population_size: 30`, `generations: 24`, `early_stop_flat_gens: 5`, `heuristic_seed_rate: 0.15`, `crossover_rate: 0.30`, `bandit_reward_mode: win_rate`, `negation_streak_` `999`, and a 3-seed cohort $\{42, 137, 271\}$. The variable knob is `EVO_HEURISTIC_FILTER`.

Reporting follows a fixed template: mean ± stderr per cell, with the 1σ-separation rule ($|\Delta| > \sigma_a + \sigma_b$) as the threshold for meaningful difference. Effective N is the count of visual-pass runs per cell.

4 Experiment Addendum

A chronological summary. Each entry is one line; methodology details appear in the run artifacts and the project journal.

1. **HG sweep, narrow genome** (2026-04-26 to 2026-04-27): the BPD/MC/RPT construction-method ordering at run-stage matches the construction ordering. BPD score 0.387 > MC score 0.285 > RPT score 0.220 on HG/binary, seed = 42.

2. **Cross-compound vs single-compound canonical cohorts on HG**: indistinguishable at $N = 3$ (cross 0.350 ± 0.083 vs BPD 0.249 ± 0.108); BPD's GA4 crossover hit rate 3.3× cross-compound's (10.42% vs 3.15%); replicates with widened genome at 1.55–3.6× across all wide-genome markets.

3. **LJ7 negation operator falsified for generative loops**: across 635 escape-mode children produced over HG and CL cohorts, 0 exceeded parent score. Default protocol disables the streak-based negate trigger (`negation_streak_threshold = 999`).

4. **Genome widening is the highest-impact change**: adding `atr14` (ATR-14 / close, scale-free) and 5 calendar features to `self.data` lifted mean Sharpe on HG/canonical from 0.46 (narrow genome, BPD-only) to ES/canonical 0.87. `atr14` appears in 24 of 25 wide-genome winners across ES, TY, ZS, GC.

5. **CL canonical cohort exposed a CAGR-blow-up attractor**: 5 of 6 runs at CAGR > 170% with Sharpe < 1.0; rolling-area metric distinguishes these via `area_std` (4–5 for visual-pass runs vs 89–99 for blow-ups).

6. **TY (10-year notes) required encoding widening**: the hardcoded `down_day` prior produced Sharpe 0.05; switching to `base_trigger = neutral` produced Sharpe 0.33 (1σ-separated on every metric). The down-day prior is a commodity/equity prior; bonds need their own.

7. **ES wide-genome canonical cohort produced highest visual-pass Sharpes** (0.687–1.020 across BPD-only and cross-compound). **ZS canonical cohort produced the highest visual-pass yield** (5 of 6 runs visually plausible). No CAGR blow-ups on either market.

8. **GC has tightest MaxDDs of any market** (−3% to −5%) but **lowest calendar-feature adoption** (`day_of_week` in 1/7 winners, vs TY's 4/6 and ZS's 3/6) — confirms gold is

macro/cross-asset driven rather than calendar driven.

9. **Auto-flag thresholds calibrated on HG/CL (narrow genome) over-fire on wide-genome cohorts**: `low_accepts < 4` fires on 5 of 6 productive ES, ZS, TY, GC runs that visually pass. The recalibration to < 3 (or rolling-ratio override at > 0.20) is supported by 4 wide-genome cohorts of evidence.

10. **Frontier-capability ablation null result** (Anthropic Claude Sonnet 4.6 vs Gemini 2.5 Flash-Lite, 5 markets, BPD seed = 42, same prompts/harness): mean Sharpe **19% lower** with Sonnet at **33× the per-run cost**. Sonnet adopts the genome more deliberately (`atr14` 5/5, `day_of_week` 5/5 vs Flash 3/5 + 1/5), but deeper feature usage does not translate to higher Sharpe. The TY case is the cleanest null: a more capable model cannot fix a wrong encoding.

5 Discussion

5.1 Encoding Dominates Capability at This Stage

The single most actionable finding is the asymmetry between encoding and capability work. Tier-1 encoding additions (one day of code, $0 marginal compute cost) lifted the mean Sharpe on HG/ES by ≈ 0.40. Substituting a frontier LLM (one week of code, $\approx$ $40 compute per ablation) produced a -0.12 mean Sharpe delta. Both interventions were measured under the same canonical protocol with paired runs.

This inverts the natural intuition. It does not say frontier LLMs are useless — Sonnet adopted the new genome columns more deliberately (`atr14` 5/5 vs Flash 3/5; `day_of_week` 5/5 vs Flash 1/5) — but *deeper feature usage did not translate to better strategy quality.* There is a layer between "knows the available primitives" and "produces a winning recombination" that frontier capability does not yet move at this scaffold's encoding granularity.

5.2 Construction-Method Ordering as Empirical Phenomenon

The BPD/MC/RPT ordering on HG (item 1 in §4) is the cleanest single-market evidence we have for the central claim of $\mathbb{H}$: that combination is operator-dependent in a way that matters downstream.

- $\oplus_{\text{bio}}$ produced an axiom set whose members are independently testable on OHLCV data. Each axiom maps to a discrete predicate (e.g., scarcity → position-sizing test; equilibrium → mean reversion; cognitive frugality → round-number anchoring). The breeder's sampling of one axiom at a time encounters a *fit local optimum* per axiom, and the GA can extend any of them.

- $\oplus_{\text{ha}}$ produced an axiom set whose members are partially defined by their relationships. The MC1/MC4 productive tension between rational pricing and cognitive frugality is an *engine for human reasoning*, but a one-axiom sample loses the arrangement. The breeder's per-axiom mutation cannot reproduce the voice's effect in isolation.

- $\oplus_{\text{inn}}$ produced an axiom set whose members are emergent abstractions ("reflexive forecast", "capital-weighted behavioral field"). They lack the OHLCV-data-availability gate that $\oplus_{\text{bio}}$ implicitly enforces during construction; the resulting axioms are aesthetically rich but operationally vacuous.

The downstream consequence is empirical: BPD's seed pool gives the GA a high-density-of-fit-axioms

search space; MC's gives a sparser one (only the orchestrated arrangement is fit, not the individual voices); RPT's gives a search space whose data-grounded fit is primarily by coincidence.

5.3 The Generative-Loop Failure of ¬

LJ7 (the negation portfolio) is a healthy operator in *judgment loops* — when a practitioner's belief set is converging on a single model of the world, ¬ probes the contrarian space and surfaces hidden assumptions. We expected the same mechanism to help in *generative loops*: when the GA's population converged on a single mutation lineage, force a few children to negate the dominant axiom to explore the contrarian space.

It did not. 635 candidate children produced via negate-mode (across HG and CL canonical cohorts) failed to exceed parent score. The mechanism is structurally different in the two contexts:

- In a judgment loop, the practitioner's parent reasoning encodes *implicit assumptions* that ¬ surfaces.
- In a generative loop, the parent code encodes *earned signal* — the output of multiple selection rounds. ¬-prompted children inherit none of that signal; they trade like fresh-from-scratch candidates in a much smaller candidate pool, and lose by selection.

This is a constraint on the *transferability* of $\mathbb{H}$ operators across contexts. Operators that earn their keep in judgment may not in generation. Each operator's contextual fitness needs separate empirical validation.

5.4 Limitations

The evidence base is small. Each canonical cohort is $3 \times 2 = 6$ runs; visual-pass rates are 25–83%, leaving effective N per cell at 0–3. Cross-config 1σ separations require effect sizes larger than the within-cell standard error, which at $N \leq 3$ is substantial. We are explicit throughout: we treat the rolling-area metric, the dud filter, and the canonical mean as *advisory* until visual scan confirms.

The Sonnet ablation at $N = 1$ is *indicative*, not settled. A defensible upgrade would be $N = 3$ paired-seeds ($\sim$ \$110 Sonnet + $\sim$ \$4 Flash). We have not performed it because the encoding-vs-capability finding is itself sufficiently large to reorient priorities toward encoding work.

The visual scan is a manual gate. Across the experiments, every candidate that passed visual scan also passed the rolling-area ratio > 0.20 threshold, but the converse is not true. The auto-flag thresholds (calibrated on HG/CL narrow-genome runs) over-fire on wide-genome cohorts; we have not yet recalibrated them against the larger cross-market dataset.

6 Future Work

The compendium and the open-work backlog jointly motivate four forward directions, in priority order:

1. `decide_entry_signal()` **widening.** The TY result motivates promoting the entry trigger from a class-attribute to a per-candidate evolved variable returning $\{-1, 0, 1\}$. Removes the asset-class prior baked into the encoding.
2. **Auto-flag threshold recalibration.** Four wide-genome cohorts (ES, TY, ZS, GC) consistently over-fire on `low_accepts < 4` and `early_plateau < 6`.

3. **Continuous-generation mode.** Static historical search produces models optimized for prior regimes; continuous generation re-evaluates the population on a rolling window with strict freeze-and-evaluate boundaries, using ↑ to update axiom credences against forward-bar performance. This shifts the project from *generating historical models* to *generating current models* and ultimately *generating future models* (forecast-conditioned candidates evaluated only when the anticipated regime arrives).

4. **Currency / interest-rate underlayment.** GC's lowest-calendar-adoption result motivates a "little-economy" macro context layer: a small set of normalized currency-strength and yield-curve signals as additional `self.data` columns. Stays within currencies and rates only — explicitly avoiding cross-market regime classification, which is curve-fitting-prone.

The Sonnet–Flash ablation null result removes "frontier-model capability" from the priority list, conditional on the encoding work not running into a regime where it stops mattering.

Addendum: Experiment Statistics (2026-04-30)

For this summary, each `results/evolution/run_*` folder that contains a `summary.json` is treated as one experiment.

Primary success criterion:

- Success: `best_fitness.score > 0`
- Failure: `best_fitness.score <= 0`

Secondary indicator (reported but not used for pass/fail): run has at least one `ACCEPTED` generation.

- Total experiments: 91
- Successes: 87
- Failures: 4
- Runs with ≥ 1 accepted generation: 85
- Overall pass rate: 95.6%

Pass-rate trend by run date (`YYYYMMDD` parsed from folder name):

Date	Success	Total	Pass Rate
20260425	9	11	81.8%
20260426	10	10	100.0%
20260427	16	16	100.0%
20260428	28	28	100.0%
20260429	24	26	92.3%

Interpretation: pass rate improves materially after the earliest day and remains high in later cohorts, consistent with harness and protocol maturation.

Addendum: Market Illustration Table (2026-04-29)

Compact illustration set requested for three "good" markets (ZS, GC, ES) and three "bad" markets (TY, CL, HG), selected from `results/evolution/run_*` artifacts.

Set	Market	Run ID	Score	Sharpe	CAGR	MaxDD
Good	ZS	run_20260429_030308_ZS	0.5464	0.830	0.069	-0.068
Good	GC	run_20260429_023604_GC	0.4300	0.644	0.037	-0.039
Good	ES	run_20260425_174650_ES	0.7705	1.190	0.063	-0.055
Bad	TY	run_20260429_141036_TY	-0.0063	-0.196	-0.002	-0.017
Bad	CL	run_20260429_121949_CL	0.4039	0.564	0.188	-0.209
Bad	HG	run_20260427_194621_HG	0.1265	0.071	0.011	-0.100

Illustration file for each row: `results/evolution/<run_id>/trades.png`.

Appendix A: Experiment Folder List (results/evolution/run_*)

run_20260425_084004_ES
run_20260425_084428_ES
run_20260425_113148_ES
run_20260425_152235_ES
run_20260425_154342_ES
run_20260425_171555_ES
run_20260425_174650_ES
run_20260425_195315_ES
run_20260425_203619_ES
run_20260425_211154_HG
run_20260425_223425_HG
run_20260426_140524_ES
run_20260426_150019_ES
run_20260426_154932_ES
run_20260426_172246_HG
run_20260426_175935_HG
run_20260426_183654_HG
run_20260426_213017_HG
run_20260426_215841_HG
run_20260426_222533_HG
run_20260426_233455_HG
run_20260427_001926_HG
run_20260427_005347_HG
run_20260427_011920_HG
run_20260427_120338_HG
run_20260427_130747_HG
run_20260427_140400_HG
run_20260427_145436_HG
run_20260427_172759_HG
run_20260427_175748_HG
run_20260427_183934_HG
run_20260427_185316_HG
run_20260427_192826_HG
run_20260427_194621_HG
run_20260427_210423_HG
run_20260427_212859_HG
run_20260427_215727_HG
run_20260428_032934_CL
run_20260428_035015_CL
run_20260428_040639_CL

```
run_20260428_043237_CL
run_20260428_052432_CL
run_20260428_055727_CL
run_20260428_162609_CL
run_20260428_165632_HG
run_20260428_170606_CL
run_20260428_173211_HG
run_20260428_174035_CL
run_20260428_181043_HG
run_20260428_184758_HG
run_20260428_191721_HG
run_20260428_195104_HG
run_20260428_200104_CL
run_20260428_204224_CL
run_20260428_213145_CL
run_20260428_214033_ES
run_20260428_220136_ES
run_20260428_222112_TY
run_20260428_222314_ES
run_20260428_223905_TY
run_20260428_225531_TY
run_20260428_230850_ES
run_20260428_232636_ES
run_20260428_234657_ES
run_20260428_235408_TY
run_20260429_001739_TY
run_20260429_004143_TY
run_20260429_014316_ZS
run_20260429_014614_GC
run_20260429_015950_ZS
run_20260429_021050_GC
run_20260429_021552_ZS
run_20260429_023604_GC
run_20260429_023820_ZS
run_20260429_025547_GC
run_20260429_030308_ZS
run_20260429_032240_ZS
run_20260429_032250_GC
run_20260429_032422_GC
run_20260429_101447_GC
run_20260429_113057_HG
run_20260429_121949_CL
run_20260429_132558_ES
run_20260429_141036_TY
run_20260429_142238_SI
run_20260429_151640_SI
run_20260429_154754_ZS
run_20260429_160948_SI
run_20260429_180703_SI
run_20260429_191846_SI
run_20260429_201057_SI
```

Appendix B: An Evolved Strategy for Soybeans

Requested source path normalized to existing artifact: `results/evolution/run_20260429_201057_SI/best_candidate.py`

```
class StrategyGen_12_25(StrategyBase):
    def decide_size(self):
        close = self.data['close']
        returns = self.data['returns']
        atr14_ratio = self.data['atr14_ratio']
        day_of_week = self.data['day_of_week']

        # Parent A's strongest gating condition: Mean reversion with confirmation
        # Condition: Price significantly below fair value AND deviation decreasing AND
    sufficient volatility AND not Friday.
        fair_value_lookback = 80
        fair_value = close.rolling(fair_value_lookback).mean().shift(1)
        deviation = close - fair_value
        abs_deviation = abs(deviation)
        deviation_std_lookback = 252
        rolling_std_deviation = deviation.rolling(deviation_std_lookback).std().shift(1)
        significant_deviation_multiplier = 2.8
        significant_deviation_threshold = significant_deviation_multiplier *
    rolling_std_deviation
        deviation_decrease_over_time = (abs_deviation.shift(1) - abs_deviation).shift(1)
        weakening_force_condition = deviation_decrease_over_time > 0
        min_atr_ratio_threshold_a = 0.01
        current_deviation_decreasing = (deviation.shift(1) - deviation) > 0

        parent_a_condition = (
            (deviation < -significant_deviation_threshold) &
            weakening_force_condition &
            current_deviation_decreasing &
            (atr14_ratio > min_atr_ratio_threshold_a) &
            (day_of_week != 4)
        )

        # Parent B's strongest gating condition: Volatility regime filtering with cost
    consideration
        # Condition: Trade in strict normal or high volatility regimes, down day, not
    Friday, sufficient ATR for costs.
        lookback_period_vol = 252
        rolling_mean_vol = atr14_ratio.rolling(lookback_period_vol).mean().shift(1)
        rolling_std_vol = atr14_ratio.rolling(lookback_period_vol).std().shift(1)
        low_vol_threshold = rolling_mean_vol - 0.10 * rolling_std_vol
        very_low_vol_regime = atr14_ratio < low_vol_threshold
        normal_vol_upper_bound = rolling_mean_vol + 0.75 * rolling_std_vol
        normal_volatility_regime = (atr14_ratio > low_vol_threshold) & (atr14_ratio <
    normal_vol_upper_bound)
        vol_85_percentile = atr14_ratio.rolling(lookback_period_vol).quantile(0.85).shift
    (1)
        high_volatility_filter = atr14_ratio > vol_85_percentile
        down_day_trigger = returns < 0
        no_friday_trade = day_of_week != 4
        min_atr_for_trade_ratio = (self.transaction_cost * 2) / close
```

```
    sufficient_atr_for_trade = atr14_ratio > min_atr_for_trade_ratio

    trade_in_normal_vol = normal_volatility_regime & down_day_trigger &
no_friday_trade & sufficient_atr_for_trade
    trade_in_high_vol = high_volatility_filter & down_day_trigger & no_friday_trade &
 sufficient_atr_for_trade
    parent_b_condition = (trade_in_normal_vol | trade_in_high_vol) & ~
very_low_vol_regime

    # Recombination Strategy:
    # We will use a regime classifier based on ATR ratio to switch between the two
parent strategies.
    # If ATR ratio is high (indicating potential for mean reversion opportunities as
in Parent A),
    # we use Parent A's logic. If ATR ratio is in a more "normal" or "high"
volatility band
    # suitable for trend-following or stronger mean-reversion plays (as in Parent B),
 we use Parent B's logic.
    # The goal is to combine the strengths: Parent A's conviction in mean-reversion
when price deviates
    # significantly, and Parent B's robust volatility regime filtering and cost
awareness.

    # Regime Classifier: Use a threshold on ATR ratio to differentiate.
    # A higher ATR ratio might indicate more opportunity for Parent A's specific mean
-reversion setup.
    # A moderate to high ATR ratio that is not extremely high might be better suited
for Parent B's regime logic.
    # Let's use a threshold that separates "very high" volatility from "normal to
high" volatility.
    # A threshold around the 85th percentile of ATR ratio could be a good switch
point.
    atr_ratio_switch_threshold = atr14_ratio.rolling(lookback_period_vol).quantile
(0.85).shift(1)

    # Condition for using Parent A's logic (e.g., when ATR ratio is very high,
suggesting strong deviations)
    use_parent_a_logic = atr14_ratio > atr_ratio_switch_threshold

    # Condition for using Parent B's logic (e.g., when ATR ratio is in a more typical
 range for its filters)
    use_parent_b_logic = ~use_parent_a_logic

    # Combine the conditions:
    # If we are in the "Parent A regime", use Parent A's condition.
    # If we are in the "Parent B regime", use Parent B's condition.
    # The final gate is 1 if EITHER of these combined conditions is met.
    # This is an OR combination of the two strategies, activated by a regime.

    gate_a = np.where(parent_a_condition, 1, 0)
    gate_b = np.where(parent_b_condition, 1, 0)

    # Apply the regime switch
    final_gate = np.where(
```

```
        use_parent_a_logic,
        gate_a,
        gate_b
    )

    # Ensure we still meet the minimum trade count requirement by potentially
loosening
    # conditions if the combined strategy is too restrictive.
    # Given the directive to combine the *strongest* conditions, we will use an AND
    # if we want to ensure both are met, or an OR if we want to leverage either.
    # The prompt suggests "combines those rules -- A's strongest condition AND B's
strongest condition".
    # This implies an AND, but the regime switching is a more nuanced way to combine.
    # Let's try an OR of the two strategies, activated by regime, which is a common
way to combine.
    # If the regime dictates Parent A, we *only* consider Parent A's gate. If regime
dictates Parent B, we *only* consider Parent B's gate.
    # This is a conditional OR.

    # Let's re-evaluate the recombination directive: "combines those rules -- A's
strongest condition AND B's strongest condition -- into a unified gate."
    # This implies a logical AND, but the regime switching is a more sophisticated
interpretation.
    # A simpler AND would be: parent_a_condition & parent_b_condition.
    # However, the regime switching is more likely to yield a Pareto improvement.
    # Let's refine the regime:
    # If ATR ratio is VERY HIGH (suggesting potential for extreme deviations that
Parent A targets): use Parent A's logic.
    # If ATR ratio is HIGH but not VERY HIGH (suggesting opportunities Parent B's
filters are designed for): use Parent B's logic.

    # Let's define "very high" ATR ratio as being above the 95th percentile.
    very_high_atr_ratio_threshold = atr14_ratio.rolling(lookback_period_vol).quantile
(0.95).shift(1)
    is_very_high_vol = atr14_ratio > very_high_atr_ratio_threshold

    # If it's very high vol, use Parent A's condition.
    # Otherwise, use Parent B's condition.
    # This is a direct switch based on volatility level.
    final_gate_logic = np.where(
        is_very_high_vol,
        parent_a_condition,
        parent_b_condition
    )

    # The prompt also mentions "The child must inherit only from StrategyBase, not
from either parent."
    # And "decide_size MUST return a pd.Series of int values in 0-or-1".
    # The "TRADE FLOOR: parent fired 2 trades, floor is 200. The harness applies a
hard score penalty proportional to shortfall -- a child with fewer than 200 trades
will be looser the entry conditions to fire more often."
    # This suggests we might need to loosen conditions if the combined strategy is
too tight.
    # However, the primary directive is to combine the *strongest* rules.
```

```
    # The current approach is a conditional combination.

    # Let's consider a simpler AND combination first, as the prompt says "AND B's
strongest condition".
    # If the number of trades is too low, we can then loosen.
    # For now, let's implement the AND of the two core conditions.
    # This ensures that any trade must satisfy both the mean-reversion conviction of
A AND the regime/cost filters of B.

    # Re-interpreting "combines those rules -- A's strongest condition AND B's
strongest condition" as a strict AND.
    # This is the most direct interpretation of "AND".
    # If this results in too few trades, we will need to revisit.
    # The "TRADE FLOOR" directive is a strong hint that we might need to loosen.
    # However, the primary goal is to combine the strongest rules.

    # Let's try a strategy that uses Parent A's logic when its conditions are met,
    # AND also requires Parent B's conditions to be met.
    # This is a strict AND.
    # This might be too restrictive and lead to very few trades.
    # The regime switching approach is generally more robust for combining strategies.


    # Given the "TRADE FLOOR" issue, a strict AND is likely to fail.
    # The regime switching is a better approach to combine strategies while
potentially increasing trade count.
    # Let's stick with the regime switching approach as it's more likely to be
productive.
    # The prompt also says "combine at the CONDITION level (logical AND, OR
conditioned on regime detection, switch between A's gate and B's gate based on a vol/
trend regime classifier, etc.)".
    # This confirms regime switching is a valid interpretation.

    # Let's refine the regime switching:
    # If ATR ratio is very high (above 95th percentile), use Parent A's condition.
    # Otherwise, use Parent B's condition. This is a direct switch.
    # This ensures that when volatility is extreme, we rely on Parent A's specific
mean-reversion setup.
    # In less extreme, but still significant volatility, we rely on Parent B's more
general regime and cost filters.

    # Final decision on recombination: Regime switching.
    # If ATR ratio is in the top 5% (very high vol), use Parent A's logic.
    # Otherwise, use Parent B's logic.

    final_gate_logic = np.where(
        is_very_high_vol,
        parent_a_condition,
        parent_b_condition
    )

    # The prompt also mentions "The child is useful only if BOTH Sharpe AND CAGR
improve versus the WORSE of the two parents on those axes".
    # This is a post-hoc evaluation criterion, not something to implement directly in
```

```
    the code logic, but it guides the choice of recombination.
        # The regime switching is more likely to find a Pareto improvement than a strict
    AND.

        # Let's ensure the output is a binary gate.
        gate = np.where(final_gate_logic, 1, 0)

        return pd.Series(gate, index=self.data.index).fillna(0).astype(int)
```


Acknowledgments

The Heuristic Algebra v0.5 spec [1] is the genome's source. The scaffold uses Anthropic's Claude family (Opus 4.7 for engineering assistance via Claude Code, Sonnet 4.6 for the frontier ablation experiment) and Google's Gemini 2.5 Flash-Lite as the default breeder. None of these models are co-authors of this paper; they are tools used in its construction and in the construction of the underlying scaffold.

References

[1] Carstens, H. (2026). *Heuristic Algebra v0.5*. https://hcarstens.github.io/assets/papers/heuristic_algebra_paper.pdf.

[2] Carstens, H. (2026). *RegimeTester* (source repository). https://github.com/hcarstens/RegimeTester.

[3] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

[4] Guo, Q., Wang, R., Guo, J., Li, B., Song, K., Tan, X., Liu, G., Bian, J., & Yang, Y. (2023). *Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers* (EvoPrompt). arXiv:2309.08532.

[5] Hemberg, E., Moskal, S., & O'Reilly, U.-M. (2024). *Evolving Code With a Large Language Model* (LLM_GP). arXiv:2401.07102.

[6] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

[7] Romera-Paredes, B. et al. (2024). *Mathematical discoveries from program search with large language models* (FunSearch). *Nature* 625, 468–475.

[8] Zhai, S. et al. (2025). *X-evolve: Evolving Solution Spaces with LLM-Generated Tunable Programs*. (preprint).

Source code, run artifacts, canonical YAML, and project journal at
https://github.com/hcarstens/RegimeTester