A Practical Theory for Symbolic Formula Discovery (Data— and Knowledge—Driven, LLM-Guided)

Cenk Alkan

October 6, 2025

Abstract

I present a complete, implementation-ready theory for discovering concise and interpretable scientific formulas from data while enforcing domain knowledge. My design is LLM-guided but decisioned by verifiable criteria. I define: a protected numeric expression language and constant policy; a canonical complexity functional with a unique normal form; an interpretability score with a calibrated logistic map; a signed, normalized multi-objective loss; deterministic decoding with seed derivation; a sound unit/dimension type system; a deterministic mutual-information (MI) feature pipeline; a duplicate-aware population schedule with a correct 2D Pareto frontier; constant fitting procedures; a small MCTS for explanations; a synthetic-data protocol preserving identifiability; and an acceptance harness that certifies equality by a symbolic zero-FN route, with refutation on a rational lattice and a protected floating probe. Every piece is specified so that implementation reduces to faithfully encoding these rules.

1 Problem Statement and Objectives

Given inputs $X \in \mathbb{R}^{n\times d}$ with variable meanings and SI dimensions, and a target $y \in \mathbb{R}^n$ (or labels), I search for $f : \mathbb{R}^d \to \mathbb{R}$ that balances accuracy, simplicity, and interpretability under unit consistency. LLMs generate candidates; I select using formal criteria:

$$\min_f \ \big(E(f),\ C(f),\ -S(f)\big),$$

where E is a normalized error, C is my complexity, and $S \in (0, 1)$ is a calibrated interpretability score. I report a 2D Pareto frontier in (C, E) and use a scalarized loss involving S only for survivor ranking.

2 Expression Language and Protected Numerics

2.1 Primitive set and guards

I restrict evaluation to a total, elementwise set of primitives with explicit guards:

$$\begin{aligned}
&\mathrm{padd}(x,y) = x + y, \quad \mathrm{psub}(x,y) = x - y, \quad \mathrm{pmul}(x,y) = xy,\\
&\mathrm{pdiv}(x,y) = \frac{x}{y^\star}, \quad y^\star = \mathrm{sign}(y)\,\max(|y|,\varepsilon),\\
&\mathrm{pneg}(x) = -x, \quad \mathrm{pabs}(x) = |x|, \quad \mathrm{plog}(x) = \log(|x| + \varepsilon_{\log}),\\
&\mathrm{pexp}(x) = \exp(\mathrm{clip}(x, -C_{\exp}, C_{\exp})), \quad \mathrm{psqrt}(x) = \sqrt{\max(x,0)},\\
&\mathrm{psin},\ \mathrm{pcos},\ \mathrm{ptanh},\ \text{and fixed powers } p \in \{-3,-2,-1,-\tfrac{1}{2},\tfrac{1}{2},2,3\}.
\end{aligned}$$

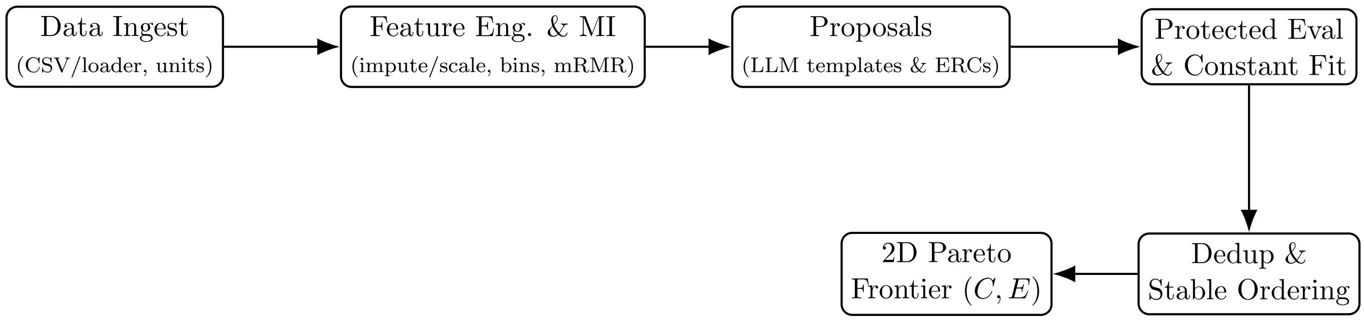

Figure 1: Canonical end-to-end process: data ingest $\rightarrow$ feature/MI $\rightarrow$ proposals $\rightarrow$ protected evaluation $\Downarrow$ dedup/ordering $\Leftarrow$ 2D Pareto frontier $\rightarrow$ acceptance (symbolic $\rightarrow$ lattice $\rightarrow$ `float_probe`) $\rightarrow$ logging/artifacts.

I treat $\{\pi, e\}$ as read-only constants. Ephemeral random constants (ERCs) appear as terminals and may be refit after structure discovery.

Proposition 1 (Well-defined objective)**.** *Any expression f built from my protected primitives is finite and Borel-measurable on $\mathbb{R}^m$. If inputs are absolutely continuous and y is finite a.s., sample losses (MSE/MAE) are finite a.s. and well-defined.*

2.2 Grammar and sandbox

I use a minimal, effect-free grammar: $\{+, -, \times, \mathrm{pdiv}, \mathrm{plog}, \mathrm{pexp}, \mathrm{psqrt}, \mathrm{psin}, \mathrm{pcos}, \mathrm{ptanh}\}$ plus the fixed powers and constants/ERCs. Static checks enforce: (i) one function body, no imports or I/O; (ii) no builtins; (iii) calls only to protected numerics. Runtime executes in a fresh process with empty builtins and a restricted numeric proxy.

Proposition 2 (Safety)**.** *Under this whitelist and sandbox, any accepted artifact can only evaluate the intended numeric expression; file/network/process effects and capability escalation are ruled out by syntax and the execution environment.*

3 Constant Policy and Fitting

I sample ERCs from a mixture: discrete mass on small rationals and $\{\pm\pi, \pm e\}$, a uniform component on $[-3, 3]$, and a signed log-uniform component. After structure discovery, I fit constants by least squares: closed-form for linear-in-parameter templates; otherwise trust-region reflective least squares with deterministic multi-starts.

Proposition 3 (Linear recovery)**.** *For noiseless $y = \Phi\theta^\star$ with full column rank, least squares returns $\hat{\theta} = \theta^\star$. With $y = \Phi\theta^\star + \varepsilon$ and $\mathbb{E}\varepsilon = 0$, $\hat{\theta} \rightarrow \theta^\star$ as $\mathrm{Var}(\varepsilon) \rightarrow 0$.*

4 Complexity Functional C

4.1 Definition and canonicalization

I parse to a canonical form and score nodes as follows. Leaves: variables cost 1; constants cost 0 for $\{0, 1\}$, cost 1 for $\{2, -1, \frac{1}{2}, \pi, e\}$, else digit-length proxy (floats simplified when exact). Internal nodes: Add/Mul cost $(k-1) + \sum C(\cdot)$; small rational/integer Pow costs $2 + C(\text{base})$; other numeric exponent costs $3 + C(\text{base})$; symbolic exponent costs $4 + C(\text{base}) + C(\text{exp})$; $\{\log, \exp, \sin, \cos, \tanh\}$ cost $3 + \sum C$; Abs costs $2 + \sum C$. Canonicalization is an explicit, finite, terminating transform set: flatten/sort Add/Mul, remove neutrals, normalize numeric factors.

Lemma 1 (Unique normal form)**.** *My transform set is semantics-preserving, terminates, and yields a unique normal form for any input expression.*

Definition 1 (Normalized complexity)**.** $C_{\min}(e) = C(\text{canon}(e))$ *where* canon *applies my fixed transform sequence.*

Proposition 4 (Subadditivity)**.** *For m-ary f and expressions $g_1, \dots, g_m$, I have $C\big(f(g_1, \dots, g_m)\big) \le C(f) + \sum_j C(g_j)$.*

5 Interpretability Score S and Calibration

I grade five criteria in $[-20, 20]$: (1) structural simplicity, (2) variable semantics/units, (3) modularity, (4) parameter interpretability, (5) domain behavior. The raw total $x \in [-100, 100]$ is mapped to $S(x) = \sigma(a(x - b))$ with $a \le 0.02$ so the global Lipschitz constant is $L = a/4 \le 0.005$.

Proposition 5 (Monotone and Lipschitz)**.** *S is strictly increasing and globally Lipschitz with constant $a/4$.*

6 Loss, Normalization, and Ranking

6.1 Metrics and normalization

For regression, $E = \text{clip}(1 - R^2, 0, 1)$ with a bounded NMAE fallback if R^2 is undefined. For classification, I use cross-entropy divided by $\log K$. I normalize E, C, S by robust min–max within a fixed batch snapshot. Let N_E, N_C, N_S denote these snapshot normalizations of E, C, S, respectively.

Definition 2 (Scalarized loss)**.** *With $(\alpha, \beta, \gamma) \ge 0$,*

$$L = \alpha\, N_E(E) + \beta\, N_C(C) - \gamma\, N_S(S).$$

Proposition 6 (Monotone in S per snapshot)**.** *For a fixed normalization snapshot, L is strictly nonincreasing in S when $\gamma > 0$.*

7 Units and Dimensional Typing

7.1 Model and rules

I model dimensions as exponent vectors $d \in \mathbb{Z}^7$ over $(M, L, T, I, \Theta, N, J)$. Multiplication/division add/subtract exponents; rational powers scale exponents; $\exp, \log, \sin, \cos, \tanh, \sqrt{\cdot}$ require dimensionless inputs; Abs preserves the input dimension.

Proposition 7 (Soundness)**.** *Given an environment $\Gamma : x \mapsto d_x$, my checker either returns a dimension d (derivation $\Gamma \vdash e : d$) or rejects with a specific rule violation. Any accepted expression is dimensionally consistent.*

8 Feature Engineering and MI Selection

My pipeline: train-only mean-imputation; standard scaling; engineered transforms (polynomial, trigs, $\log(|x| + \varepsilon)$, pairwise products); plug-in MI I_b with equal-width bins ($b = 16$); a stable top-K prefilter and a greedy mRMR-style diversity selection with a fixed correlation cutoff.

Proposition 8 (Consistency for discretized MI)**.** *With fixed finite partitions and i.i.d. samples, the plug-in $I_b(X;Y)$ converges almost surely to $I(X_b;Y_b)$, the MI between discretized variables.*

9 Population Schedule, Deduplication, and Budget

I use a steady-state loop with typical settings $N{=}200$ initial, survivor cap $K{=}30$, new per round $J{=}10$, horizon $T{=}500$. I control duplicates by canonical structural keys, numeric fingerprints on a fixed grid, and (when needed) a symbolic certificate; survivors are ordered by $(L, \text{canonical form})$ stably.

Proposition 9 (Partial order)**.** *Define $f \preceq g$ if either their canonical forms match, or numeric fingerprints match and* $\text{simplify}(f - g) = 0$ *and $C_{\min}(f) \le C_{\min}(g)$. Then $\preceq$ is a partial order; antisymmetry holds on canonical representatives.*

10 Pareto Frontier in Two Dimensions

For (C, E) minimization with tolerance $\varepsilon \ge 0$, I sort stably by (C, E) and sweep left-to-right, accepting a point iff $E < E^\star - \varepsilon$ (then updating $E^\star$). This returns exactly the ε-nondominated set in $O(n \log n)$.

Proposition 10 (Correctness)**.** *The sweep returns all and only ε-nondominated points (with stable tie handling for duplicates).*

11 Splits, Seeds, and Leakage Control

I use seeded stratified holdouts and K-folds. For grouped entities, I ensure group-disjoint splits. All partitions are deterministic in the seed, and my group-aware procedures eliminate entity leakage by construction.

Proposition 11 (Stratified holdout bias)**.** *Under exchangeability and classwise sampling without replacement, the stratified holdout estimate equals the class mixture of class risks; rounding effects induce $O(1/N)$ bias.*

12 Deterministic Decoding and Prompt Discipline

12.1 Backend, endpoint, and model

Unless otherwise noted, all completions are generated by **llama3.1** served through the **Ollama** HTTP API at `http://127.0.0.1:11434`. The implementation treats this endpoint as the default local backend; if an environment override is provided, it is recorded in the run manifest. When available, the model digest reported by Ollama is persisted alongside `model_id` to pin the exact binary.

12.2 Seed derivation and knobs

I fix decoding knobs (temperature, top-k, top-p, repetition penalty, token budget) and derive a 64-bit seed by

$$\mathrm{seed}(s, p, i, a) = \mathrm{uint64}\big(\mathrm{SHA256}(s\|p\|i\|a)\big),$$

where s is the session ID, p a prompt tag, i an item index, and a a retry counter. Under *llama3.1* via Ollama, these knobs and the seed are passed deterministically through the client so that, given identical prompts and the same model digest, the sampled completions are reproducible.

Proposition 12 (Determinism)**.** *If base logits are deterministic in $(p, y_{<t})$ at the backend and the only randomness comes from my seeded PRNG, completions are deterministic in (s, p, i) (and retry schedule).*

12.3 Prompt contract

I constrain LLM outputs to a single fenced Python function `def f(env):` with no imports and only whitelisted names. I enforce stop sequences so I always extract exactly one artifact per sample.

13 Explanation Search via MCTS

I include a small MCTS with UCB1 selection ($c = \sqrt{2}$), bounded rollouts, and fixed budgets to generate human-facing explanations. Rewards are bounded in $[0, 1]$ and distinct from S.

Remark 1 (Use in practice)**.** *I use UCB1 as a robust exploration–exploitation policy; in nonstationary trees it is a practical heuristic, not a formal optimality claim.*

14 Synthetic Data and Identifiability

For rediscovery tests, I sample over safe domains, affine-normalize features into $[0, 1]^d$, and (optionally) add Gaussian noise to normalized targets. Success means the planted form lies on my (C, E) frontier.

Proposition 13 (Normalization preserves class)**.** *If $u_j = (x_j - L_j)/(U_j - L_j)$ are strictly increasing affine maps and the hypothesis class is closed under such reparameterizations, normalization preserves representability and structural identifiability (constants rescale).*

15 Acceptance Criteria and Certification

I canonicalize f (truth) and g (candidate) and apply a three-stage contract:

1. **Symbolic certificate (zero-FN).** If $\mathrm{simplify}(f - g) = 0$, I accept with method `symbolic`.

2. **Deterministic rational-lattice refutation.** If symbolic fails, I search a fixed rational lattice in exact arithmetic. A finite, non-singular witness implies $f \neq g$ and I reject with method `reject`.

3. **Protected floating probe.** If the lattice finds no witness, I probe on a safe float grid at tight tolerance and compute a simple miss-probability bound. Passing probes accept with method `float_probe`.

Proposition 14 (Determinism and soundness)**.** *For fixed (f, g), variable order, and horizon, the lattice is deterministic; any returned witness is a certified counterexample. The symbolic route has zero false negatives. Absent both, a float probe with m i.i.d. points misses a region of disagreement of relative measure $\delta > 0$ with probability at most $e^{-\delta m}$.*

16 Reproducibility and Logging

I record a manifest (decoding knobs, seed, environment) and canonical JSONL events. The manifest includes the LLM binding:

$$\begin{aligned}&\{\texttt{model\_id} = \text{"llama3.1"},\ \texttt{backend} = \text{"ollama"},\\ &\ \texttt{endpoint} = \text{"http://127.0.0.1:11434"},\\ &\ \texttt{model\_digest} = \text{(if available)}\}.\end{aligned}$$

The manifest hash is the SHA-256 of a canonical JSON core; event timestamps are logical and monotone. For each accepted/rejected pair I persist a compact artifact (`accept_proof.json`) and the canonical forms (`forms.txt`).

17 Scope and Limitations

Token and time budgets limit large-scale exploration. My acceptance is deliberately conservative: exact symbolic equalities are decisive; the lattice is refutation-only; the float probe is tightly guarded. I can layer broader proposal sets, hierarchical search, or retrieval without changing these foundations.

Conclusion

I fixed a protected numeric language and constant policy, a canonical complexity, a calibrated interpretability score, a signed and normalized loss, deterministic LLM decoding, a sound unit system, a deterministic MI pipeline, a duplicate-aware population schedule with a provably correct 2D frontier, an MCTS explainer, a synthetic protocol, and strict acceptance criteria. The design is implementation-ready and empirically grounded: the behavior I report in results follows from these contracts rather than ad hoc choices.