L2G-SR: LEARNING TO GUIDE LLM-BASED SYMBOLIC REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic regression searches for an equation that explains observed data. Numerical error is commonly used to guide the search, but it alone can favor an accurate surrogate over a candidate with the right structure. We propose L2G-SR, a framework for LLM-based symbolic regression that guides search with a learned structural verifier. We construct structural supervision signal using tree edit distance (TED), accounting for commutative operators and alternative expression forms. A verifier first aligns numerical observations with the formulas, then learns to score a candidate's structural similarity to the target formula. At search time, the target formula is hidden: the verifier complements numerical error in parent selection. An analyst then uses verifier feedback and diagnostic tools to propose revisions, which a generator converts into refined expressions. On LLM-SRBench, L2G-SR achieves state-of-the-art numerical fit, equation level accuracy and TED across the evaluated LLM backbones and domains. L2G-SR demonstrates the value of learned structural guidance for LLM-based symbolic regression.

1 INTRODUCTION

Symbolic regression discovers both the structure and constants of an expression that explains observed variables, making it useful for interpretable modeling and the discovery of candidate scientific laws (Schmidt & Lipson, 2009; Cranmer, 2023). However, finding an optimal expression is NP-hard in general (Virgolin & Pissis, 2022), and the combinatorial growth of possible structures makes efficient exploration essential. Existing approaches include search-based methods that evolve candidate expressions (Koza, 1992; Cranmer, 2023), neural methods that learn to map observations to symbolic expressions (Biggio et al., 2021; Kamienny et al., 2022), and methods based on large language models (LLMs) that combine scientific priors with numerical feedback to propose and refine equations (Shojaee et al., 2025a; Grayeli et al., 2024; Wang et al., 2025; Pang et al., 2026; Li & Pan, 2026).

Despite these advances, existing methods still rely heavily on numerical fit to guide search and evaluate recovered formulas using metrics such as normalized mean squared error (NMSE) and accuracy at a 1% relative-error tolerance ($\mathrm{Acc}_{0.01}$). It is essential feedback, but it need not identify the most useful structure to revise: near the origin, $x - x^3/6$ approximates $\sin x$ despite using different operators, while $\exp(1.05x)$ shares the form of $\exp(x)$ yet can accumulate substantial error. Numerical metrics alone are therefore insufficient to reliably guide formula discovery, motivating a complementary structural score.

To address this gap, we introduce L2G-SR through three connected components, which guides LLM-based SR with a learned structural verifier (Figure 1). A *modified tree edit distance* (TED) abstracts constants, matches commutative operands, and compares selected algebraic views to label structural similarity. A *learned verifier* first aligns observations with their generating formulas, then learns to predict a candidate's structural similarity. A *two-signal search* combines verifier output and numerical error to select parents, suggest edits, and retain candidates.

On 240 LLM-SRBench equations, L2G-SR achieves the highest or the second highest performance on all metrics and LLM backbones, showing the strong performance of our method. Replacing its structural channel with validation MSE substantially lowers the performance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

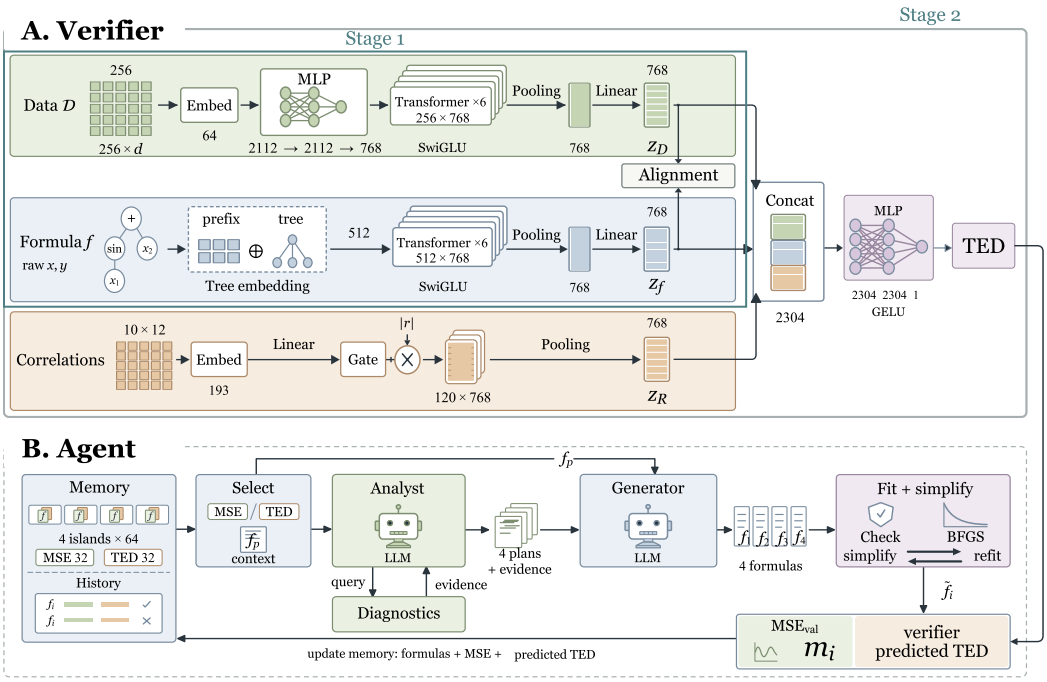


Figure 1: Overview of L2G-SR. (A) A verifier learns TED with two-stage training. (B) During search, numerical error and predicted TED guide candidate selection and retention.

Our contributions are threefold:

- We propose a modified TED that measures structural similarity between formulas while accommodating common simplifications and equivalent forms, providing structural supervision beyond numerical fit alone.
- We develop a multimodal TED verifier that predicts a candidate's structural distance to an unseen target formula from the candidate and its observations, making structural feedback available during symbolic regression.
- We integrate this learned feedback into an LLM-guided search agent, enabling it to prioritize and revise formulas that are structurally promising as well as numerically accurate.

2 Related Work

Symbolic and neural search. Genetic programming, Eureqa, Operon, PySR, GP-GOMEA, AI Feynman, and Bayesian SR search expressions through evolution, physical structure, or probabilistic modeling (Koza, 1992; Schmidt & Lipson, 2009; Burlacu et al., 2020; Cranmer, 2023; Virgolin et al., 2021; Udrescu & Tegmark, 2020; Udrescu et al., 2020; Jin et al., 2019); SRBench compares representative methods (La Cava et al., 2021). DSR and uDSR learn generation policies, while neural-guided GP, DGSR-MCTS, and TPSR combine learned proposals with search (Petersen et al., 2021; Landajuela et al., 2022; Mundhenk et al., 2021; Kamienny et al., 2023; Shojaee et al., 2023). Pretrained and end-to-end models amortize discovery over synthetic corpora (Biggio et al., 2021; Kamienny et al., 2022; Holt et al., 2023). SNIP, MDLformer, and MOD-SR connect numerical and symbolic representations (Meidani et al., 2024; Yu et al., 2025; Xiang et al., 2026). We use such representations to score a candidate's structural similarity for an external search procedure.

LLM-guided discovery and structural comparison. LLM-SR, LaSR, and DrSR use language-model priors to refine expressions (Shojaee et al., 2025a; Grayeli et al., 2024; Wang et al., 2025). DE, RESTART, SR-Scientist, and related systems add diagnostics, reusable structures, multiple

roles, or engine control (Pang et al., 2026; Li & Pan, 2026; Xia et al., 2026; Zhao et al., 2026; Saveliev et al., 2026; Xie et al., 2026; Ma et al., 2024). Our distinction is the supervised structural signal in selection, diagnosis, and retention. TED compares labeled trees (Zhang & Shasha, 1989); SRSD and PHYBench evaluate recovered structure, and EGG-SR represents equivalences with equality graphs (Matsubara et al., 2024; Qiu et al., 2025; Jiang et al., 2026).

3 METHOD

3.1 PROBLEM FORMULATION

Given observations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ and $y_i = f^\star(x_i) + \epsilon_i$, symbolic regression seeks a compact expression f for the unknown target $f^\star$. The admissible expression space $\mathcal{F}$ is defined by the available variables, operators, and constants; each candidate has a symbolic structure and fitted numerical constants. Numerical fit is commonly measured by mean squared error,

$$\mathcal{L}_{\mathrm{MSE}}(f; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \big(f(x_i) - y_i\big)^2. \tag{1}$$

Finite observations can support expressions with different structures, so low empirical error alone does not establish similarity to $f^\star$. We therefore seek candidates with good numerical fit while estimating their structural similarity to the target from $\mathcal{D}$ and the candidate expression. The target expression remains hidden during search.

3.2 STRUCTURAL SIMILARITY MEASURE

We use a modified tree edit distance (TED), $d_{\mathrm{str}}(f, g)$, to compare formula structures despite fitted constants, operand order, and common algebraic rewrites; smaller values indicate closer structures.

Step 1: Abstract trees and edit costs. To ignore constant values and grouping, we map numeric-only subtrees to `number`, symbolic constants to `param`, and flatten Add/Mul nodes. Insertions, deletions, and other relabelings cost one; identical labels and `number`/`param` exchanges cost zero (Zhang & Shasha, 1989). With PHYBench's discount (Qiu et al., 2025), insertion or deletion of a subtree of size s costs

$$\kappa(s) = \min\{s,\ 0.6(s-5) + 5\}. \tag{2}$$

Step 2: Match commutative operands. Let $D_{\mathrm{ord}}(A, B)$ be ordered TED under these costs. At addition or multiplication roots, we also match children by minimum-cost assignment to reduce sensitivity to operand order:

$$D_{\mathrm{hyb}}(A, B) = \begin{cases} \min\big\{D_{\mathrm{ord}}(A, B),\ c(a, b) + \min_\pi \sum_{i=1}^{m} C_{i,\pi(i)}\big\}, & a, b \in \{\mathrm{Add}, \mathrm{Mul}\}, \\ D_{\mathrm{ord}}(A, B), & \text{otherwise.} \end{cases} \tag{3}$$

Here a, b are root labels, c is their relabeling cost, and m is the larger child count. C_{ij} is the recursive hybrid cost for real children or $\kappa(s)$ for a dummy match. We retain the cheaper of ordered editing and child assignment.

Step 3: Compare algebraic views. We compare three views to tolerate common rewrites: a structure-preserving candidate parse (evaluated reference), ordinary evaluation with constant absorption, and expansion with power simplification. Let P_k^c, P_k^r transform the candidate and reference, and T build the tree:

$$d_{\mathrm{str}}(f, g) = \min_{k \in \{0,1,2\}} \frac{D_{\mathrm{hyb}}(T(P_k^c(f)), T(P_k^r(g)))}{|T(P_k^r(g))|}. \tag{4}$$

Each cost is relative to the reference-tree size. Appendix A specifies the transformations.

Figure 2 illustrates the shared abstract representation and the distinctions retained by this measure. More examples and explanations are provided in Appendix A.

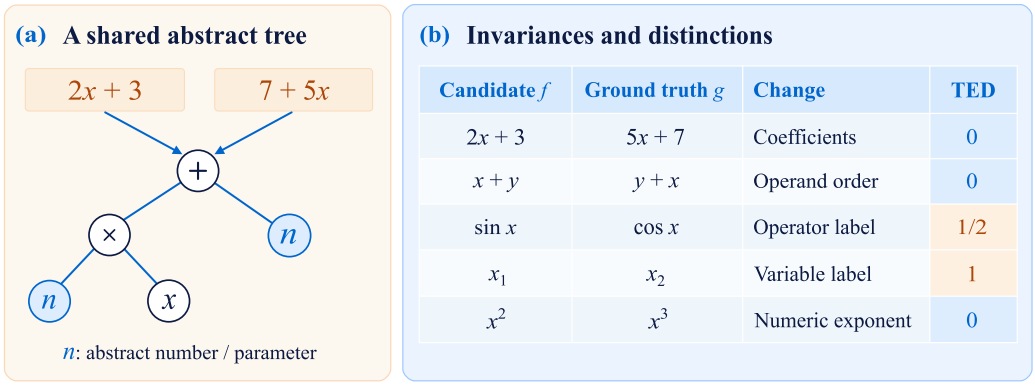


Figure 2: Examples of structural similarity under the modified TED. (a) The two linear expressions map to the same abstract tree. (b) Scores are normalized by the ground-truth tree size. In these examples, changes in coefficient values, operand order, and numeric exponent values yield zero distance, whereas operator and variable label changes incur a cost.

3.3 Data Generation

Training our Transformer verifier requires a large volume of paired numerical and symbolic data. To produce diverse synthetic formulas that resemble scientific equations and search-time expressions, we use a fixed data-generation scheme informed by SRBench/Feynman ground-truth formulas and recorded LLM-SR/PySR search trajectories on SRBench. These sources provide realistic structural priors, while all training instances are independently generated by our own synthesis algorithm, without access to LLMSRbench. The resulting data pools contain 91.9 million target–candidate pairs. To prevent benchmark overlap, we remove training pairs that share the same symbolic structure with any LLM-SRBench ground-truth expression after constant abstraction and symbolic canonicalization. More details are provided in Appendix B.

Random formulas. We first sample the total number of nodes and operators from empirical priors. We then sample an ordered tree shape using Catalan weights and place the operators, variables, and constants. These priors balance ground-truth and LLM-SR search formulas, preserving common structures while generating diverse new expressions.

Target–candidate pairs. We construct two types of target–candidate pairs: random and edited. Random pairs draw a candidate f independently of the target $f^\star$; edited pairs obtain f by locally modifying $f^\star$, including point relabeling, unary insertion or deletion, subtree replacement, etc. Random pairs mainly generate distant pairs (TED ≥ 0.6), whereas edited pairs mainly generate close pairs (TED < 0.6). Our aim is to cover as broad a range of TED values as possible. To reduce bias from the uneven TED distribution of generated pairs, we sample 12 TED intervals equally.

Simplification and validation. We fold constant-only subtrees and apply selected algebraic simplifications. We reject scientifically implausible formulas, such as those with atypical operator nesting or invalid domains. This filtering aims to keep synthetic formulas scientifically plausible and closer to the structural patterns of LLM-generated search expressions.

Numerical observations. We adapt the observation sampler of MDLformer (Yu et al., 2025) to generate observation points that cover diverse regions of the input space. For each synthetic target $f^\star$, we draw inputs from a mixture of randomly located and rotated Gaussian or uniform components. The resulting observation set is

$$\begin{aligned} &\mathcal{D}^\star = \{(\mathbf{x}_i, f^\star(\mathbf{x}_i))\}_{i=1}^{M}, && \mathbf{x}_i \sim \sum_{k=1}^{C} \rho_k Q_k, && Q_k \in \{Q_k^{\mathrm{G}}, Q_k^{\mathrm{U}}\}, \\ &Q_k^{\mathrm{G}} = \mathrm{Law}((\boldsymbol{\mu}_k + \boldsymbol{\epsilon}_k)R_k)\,, && \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\mathbf{c}_k))\,, \\ &Q_k^{\mathrm{U}} = \mathrm{Law}((\boldsymbol{\mu}_k + \sqrt{\mathbf{c}_k} \odot \mathbf{u}_k)R_k)\,, && \mathbf{u}_k \sim \mathcal{U}([-1,1]^d). \end{aligned} \tag{5}$$

Here, R_k is a random orthogonal rotation; before rotation, the mean is $\boldsymbol{\mu}_k$ and coordinatewise variances are $\mathbf{c}_k$ (Gaussian) or $\mathbf{c}_k/3$ (uniform).

3.4 VERIFIER

Architecture.

The verifier first aligns numerical observations $\mathcal{D}$ with their generating formulas in a shared embedding space. It then combines the observation embedding with a candidate formula embedding and regression summaries $R(\mathcal{D})$ to estimate the candidate's TED to the unknown target:

$$z_D = E_D(\mathcal{D}), \qquad z_f = E_f(f), \qquad z_R = E_R(R(\mathcal{D})), \qquad \widehat{d} = h_\theta([z_D; z_f; z_R]). \tag{6}$$

Embedding. Following MDLformer (Yu et al., 2025), we encode each input and target value in scientific notation using three tokens: its sign (positive or negative), its mantissa (the coefficient rounded to four significant digits), and its base-10 exponent (which specifies the power of ten multiplying the coefficient); for example, $54.321 \approx 5.432 \times 10^1 \longmapsto [+, 5.432, +1]$. As in MDLformer (Yu et al., 2025), we tokenize f in prefix order. However, TED prediction relies on the expression's tree structure, while prefix order does not explicitly mark each token's depth or branch. We therefore add a tree address to each token (Scarlatos & Lan, 2023). For a root-to-node path with child slots $(c_1, \ldots, c_d)$, its address contribution is

$$a = W_A \frac{\sum_{\ell=1}^{d} e_{\text{level}}(\ell) \odot e_{\text{slot}}(c_\ell)}{\sqrt{\max(d, 1)}} + e_{\text{depth}}(d). \tag{7}$$

At each depth, the level and child-slot embeddings together record the branch taken; adding these records describes the path from root to node, scaling limits the effect of path length, and W_A places the result in the token-embedding space while $e_{\text{depth}}(d)$ marks its depth.

Regression summaries. For each variable, we evaluate 12 simple relations, including $\log y \sim \log x_j$, $y \sim x_j$, and $y \sim x_j^2$ (Appendix C). Let r be the Pearson correlation between the transformed input and target values. Each valid relation is encoded as

$$\mathbf{t} = |r| \ \text{Linear}([\mathbf{e}_{\text{relation}}; r]).$$

Here $\mathbf{e}_{\text{relation}}$ joins three learned vectors that specify whether the output is y or $\log y$, the operation on the input (such as x, x^2, or $\log x$), and which input variable x_j is used. The $|r|$ factor downweights weak correlations. The log–log relation probes how input and output magnitudes scale together; the other 11 probe different input–output relations, such as linear and quadratic patterns.

Two-Stage Training.

Stage 1: symbolic–numeric alignment. Following MDLformer (Yu et al., 2025), which builds on SNIP (Meidani et al., 2024), we use symmetric in-batch contrastive learning to align $u_i = E_D(\mathcal{D}_i)$ with $v_i = E_f(f_i^\star)$, so observations and their generating formulas have consistent embeddings:

$$\mathcal{L}_{\text{align}} = -\frac{1}{2B} \sum_{i=1}^{B} \left[\log \frac{\exp(u_i^\top v_i / t)}{\sum_{j=1}^{B} \exp(u_i^\top v_j / t)} + \log \frac{\exp(v_i^\top u_i / t)}{\sum_{j=1}^{B} \exp(v_i^\top u_j / t)} \right], \tag{8}$$

Stage 2: structural prediction. Starting from the Stage 1 encoders, we train the verifier to predict normalized TED $d = d_{\text{str}}(f, f^\star)$ for observation–candidate pairs using

$$\mathcal{L}(\widehat{d}, d) = \begin{cases} |\widehat{d} - d|, & d \le 1, \\ \max(0, 1 - \widehat{d}), & d > 1. \end{cases} \tag{9}$$

For $d > 1$, candidates are already structurally far from the target, so we only require $\widehat{d} \ge 1$. This limits the effect of large distances and helps keep predictions from clustering near the mean. Appendix C gives the training configuration and checkpoint selection.

3.5 L2G-SR: TED-GUIDED SYMBOLIC SEARCH

We present L2G-SR, a symbolic regression method that finds formulas by repeatedly proposing and testing candidates (Cranmer, 2023; Shojaee et al., 2025a). Its core principle is to guide the

search with two complementary signals: numerical fit to the observations and predicted TED to the unknown target formula. An LLM analyst suggests revisions, and an LLM generator writes new formulas. Further details are provided in Appendix D.

We compute validation MSE on a fixed 20% holdout of the search observations to measure each candidate's predictive accuracy beyond the fitting subset.

We begin by generating formulas for four islands, each a separate candidate pool. In each search round, we select an island and a parent formula using either validation MSE or predicted TED. The analyst examines the parent and suggests four distinct ways to revise it; the generator then turns these suggestions into four complete formulas. We check each formula, fit its constants, simplify it, and score it with both signals before updating the selected island. After the search, we choose a simple formula among those with the best numerical fit. Algorithm 1 summarizes the loop.

Dual-score selection. Each round considers one of the four islands. We randomly draw a few formulas from that island, then choose one to revise (the parent): half the rounds take the formula with the lowest validation MSE, and the other half take the one with the lowest predicted TED. If $\mathcal{S}$ denotes the formulas drawn in the current round, then

$$f_{\text{parent}} = \begin{cases} \arg\min_{f \in \mathcal{S}} \text{MSE}_{\text{val}}(f), & \text{MSE round}, \\ \arg\min_{f \in \mathcal{S}} \widehat{d}(f), & \text{TED round}. \end{cases} \tag{10}$$

Analyst agent. The analyst receives validation MSE and predicted TED for the parent and several retained formulas, plus summaries of the parent's residuals and transformed-variable regressions. In TED rounds, it compares lower-TED formulas with the parent to spot terms or operators worth testing. In MSE rounds, it focuses on patterns in the residual $y - f(x)$. To check a specific idea, it can name a reference formula in a structure-comparison request or a parent subtree in a residual-probe request. The first tool reports which parts differ; the second tests whether the selected subtree explains the residual on training data. Using these results and past edit outcomes, the analyst gives the generator four specific suggestions about what to change or keep.

Generator agent. A second LLM, the generator, uses the parent, the analyst's suggestions, and cited evidence to produce four complete formulas. Each formula occupies one proposal slot and is checked, fitted, and scored before it can be kept for later rounds. The agent is told the physical units and restrictions needed for formulas to remain real-valued.

Fitting and simplification. For each generated formula, we keep its symbolic structure fixed and use BFGS to optimize any unknown constants on the fitting subset.

We use SymPy to simplify algebraically equivalent expressions and combine redundant constants. We also remove terms that contribute little to the formula's predictions, as defined by Equation 26 in Appendix D. Such terms create spurious branches in the formula tree, which later revisions may inherit or modify and thus steer the search away from the useful structure. They can also make the verifier predict an excessively large TED for an otherwise promising formula. We replace exponents that are already close to simple values. We optimize the remaining constants again with BFGS after removing terms or replacing exponents. The verifier scores only the final accepted formula.

Memory. We keep four islands, each with at most 64 formulas, plus a separate history of proposals and outcomes for the analyst. After each round, valid formulas enter the selected island, and we compute each member's island-local rank by validation MSE and by predicted TED, with lower scores ranked first in both. When formulas share the same structure, only the version with lower validation MSE enters these rankings. If an island exceeds 64 formulas, we prune it using the two rankings. We keep up to 32 leading formulas from each ranking. The survivors can become parents in later rounds; final selection favors simple formulas among those with the best validation fit.

4 Experiments

4.1 Experimental setup

Benchmark. We evaluate on LLM-SRBench (Shojaee et al., 2025b), a symbolic regression benchmark designed to test discovery beyond memorized canonical formulas. Our evaluated configuration

contains 240 equations across five domains: 111 Transform, 44 Physics, 25 Materials, 36 Chemistry, and 24 Biology.

Baselines. We compare L2G-SR with representative LLM-based symbolic regression methods, including LLMDirect (Shojaee et al., 2025b), LLM-SR (Shojaee et al., 2025a), LaSR (Grayeli et al., 2024), SGA (Ma et al., 2024), DE (Pang et al., 2026), A-SR (Zhao et al., 2026), RESTART (Li & Pan, 2026), and SR-Scientist (Xia et al., 2026).

Metrics. We report normalized mean squared error (NMSE; lower is better), accuracy at a relative-error tolerance of 0.01 ($\mathrm{Acc}_{0.01}$; higher is better), and true normalized tree edit distance (TED; lower is better) between a selected formula and the target expression. Following DE's evaluation protocol (Pang et al., 2026), we score test predictions in float64 and discard 5% of the observations with the largest relative errors for each equation. For the retained set I and its target mean $\bar{y}_I$, the numerical metrics are

$$\mathrm{NMSE}(f) = \frac{\sum_{i\in I}(f(x_i)-y_i)^2}{\sum_{i\in I}(y_i-\bar{y}_I)^2}, \qquad A(f) = \mathbf{1}\left[\max_{i\in I}\frac{|f(x_i)-y_i|}{\max(|y_i|,10^{-12})} \le 0.01\right]. \tag{11}$$

We average equation-level NMSE, $A(f)$, and TED within each domain.

4.2 Main experimental results

Table 1: Performance on five LLM-SRBench domains with Llama3.1-8B-Instruct and Qwen3-4B-Instruct-2507. Columns report NMSE (↓) and $\mathrm{Acc}_{0.01}$ (↑, %). LLMDirect, LLM-SR, LaSR, SGA, and DE results are from DE (Pang et al., 2026); A-SR results are from Zhao et al. (2026). L2G-SR shows mean ± sample standard deviation over three seeds. Bold and underlined means mark the best and second-best displayed values.

	Transform		Physics		Materials		Chemistry		Biology	
Method	NMSE ↓	$\mathrm{Acc}_{0.01}$ ↑	NMSE ↓	$\mathrm{Acc}_{0.01}$ ↑	NMSE ↓	$\mathrm{Acc}_{0.01}$ ↑	NMSE ↓	$\mathrm{Acc}_{0.01}$ ↑	NMSE ↓	$\mathrm{Acc}_{0.01}$ ↑
Llama3.1-8B-Instruct										
LLMDirect	2.95e-1	34.23	9.95e-3	0.00	8.16e-2	24.00	1.22e0	2.78	1.29e-1	4.17
LLM-SR	2.42e-1	34.23	3.00e-3	6.82	2.16e-1	60.00	5.24e-2	16.67	1.76e-2	12.50
LaSR	2.62e-1	33.33	6.07e-3	9.09	9.47e-4	32.00	1.82e-3	8.33	6.40e-3	0.00
SGA	3.52e-1	0.90	1.55e-1	2.27	4.35e-2	12.00	4.58e-2	8.33	2.42e-1	0.00
DE	1.12e-1	36.04	1.01e-3	11.36	2.89e-4	64.00	4.16e-4	11.11	1.17e-2	16.67
A-SR	**1.09e-1**	40.54	9.03e-4	27.27	7.45e-6	84.00	2.60e-5	31.94	**5.38e-6**	50.00
RESTART	2.22e-1	4.80	7.48e-3	6.06	8.19e-5	66.67	2.24e-6	60.19	1.80e-2	22.22
SR-Scientist	3.03e-1	3.00	3.78e-3	23.48	1.77e-3	52.00	4.96e-5	63.89	7.26e-2	20.83
L2G-SR	1.12e-1	**43.54**	**2.70e-4**	**89.39**	**1.07e-7**	**100.00**	**1.23e-6**	**85.05**	6.06e-3	**68.06**
	±0.04e-1	±1.04	±1.11e-4	±3.47	±0.23e-7	±0.00	±0.67e-6	±1.50	±4.05e-3	±6.36
Qwen3-4B-Instruct-2507										
LLMDirect	3.55e-1	24.32	5.46e-2	6.82	1.42e-3	52.00	2.66e-1	2.78	4.46e-2	8.33
LLM-SR	3.15e-1	26.13	2.51e-3	6.82	3.55e-3	44.00	3.36e-2	13.89	1.88e-2	12.50
LaSR	1.83e-1	30.91	6.04e-3	6.82	6.21e-4	8.00	2.31e-3	2.78	9.56e-3	0.00
SGA	4.09e-1	19.81	1.04e-1	0.00	1.02e-2	16.00	1.61e-1	2.78	1.73e-1	4.17
DE	**1.15e-1**	**50.45**	4.37e-4	15.91	1.47e-4	56.00	1.88e-4	13.89	6.69e-3	12.50
A-SR	1.37e-1	33.33	6.68e-4	21.59	3.48e-5	68.00	1.27e-5	34.72	**1.25e-3**	20.83
RESTART	6.25e1	1.50	1.86e-1	3.03	5.92e-3	4.00	6.95e-2	24.07	1.78e0	9.72
SR-Scientist	2.50e-1	19.82	2.90e-2	15.91	1.08e-3	64.00	6.50e-6	61.11	9.44e-2	41.67
L2G-SR	1.18e-1	42.94	**3.49e-4**	**87.88**	**1.20e-7**	**98.67**	**1.34e-6**	**79.63**	3.97e-3	**61.11**
	±0.08e-1	±0.52	±1.12e-4	±2.62	±0.15e-7	±2.31	±0.18e-6	±4.24	±1.28e-3	±2.41

Research Question 1: How accurately does L2G-SR fit benchmark equations? Across both backbones and all five LLM-SRBench domains, L2G-SR achieves state-of-the-art performance in numerical fit and equation-level accuracy (Table 1). Its NMSE and $\mathrm{Acc}_{0.01}$ rank first or second for every backbone and domain, showing consistent performance across models and domains. In the available final-formula comparisons, L2G-SR also attains the lowest mean true TED in every domain under both backbones (Figure 3), indicating that it finds formulas structurally closer to the targets. Additional RQ1 results, including baseline variability and LLM usage, appear in Appendix E (Tables 16 and 17).

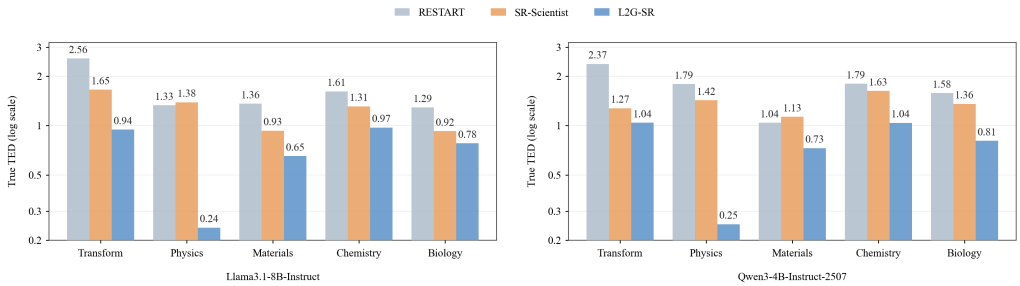


Figure 3: True TED (↓) of the final formulas from Table 1.

4.3 Verifier analysis

Research Question 2: How well does the verifier perform?

Stage 2 random test set. Table 2 reports metrics on the random test set for the checkpoint used in search. The checkpoint performs strongly on this random dataset, with accurate estimates for structurally close formulas and useful detection of more distant ones.

Metric	Selected
Val. objective ↓	0.12613
Val. $\text{MAE}_{\leq 1}$ ↓	0.13086
Val. $\text{recall}_{>1}$ ↑ (%)	52.63

Table 2: Selected Stage 2 checkpoint on the synthetic validation set. Objective follows Eq. (9). $\text{MAE}_{\leq 1}$ is the mean absolute error on targets with TED ≤ 1; $\text{Recall}_{>1}$ is the percentage of targets with TED > 1 predicted at or above 1.

Domain	MSE ↓	Correlation
Transform	0.4550	0.3836
Physics	0.1107	0.3310
Materials	0.1148	0.2505
Chemistry	0.1393	0.5808
Biology	0.1000	0.4318

Table 3: Verifier performance measured after search on recorded candidate formulas. Correlation is Pearson r. Evaluation details appear in Appendix E.7.

Downstream predictions. On recorded search candidates, the verifier predicts TED with reasonable accuracy (Table 3), providing effective structural guidance that improves downstream search (Table 4).

The matched Physics ablation gives more direct evidence for the verifier's role: replacing its score with validation MSE in the structural channel worsens test NMSE and accuracy while increasing the final formula's true TED (Table 4).

4.4 Ablation study

Research Question 3: Does TED feedback improve search under a matched budget? We ablate the TED verifier on Physics by replacing its predicted TED score with internal-validation MSE throughout the structural channel. We keep two-role generation, diagnostics, fitting and simplification, the selection schedule, retention policy, and the 400-slot budget fixed, so the comparison isolates the verifier's structural guidance. As shown in Table 4, removing this guidance substantially worsens both NMSE and accuracy, demonstrating that the TED verifier is an important contributor to L2G-SR's search performance in this setting.

Notably, this downstream benefit is observed despite moderate and domain-dependent candidate-level verifier correlations (Table 3). Together, Tables 3 and 4 suggest that the verifier's value may lie less in precisely estimating absolute TED for every candidate than in providing sufficiently informative relative structural preferences during search.

Research Question 4: Which verifier inputs and training stages improve prediction?

We compare the full verifier with four variants: (i) *without tree address*, which zeroes the formula encoder's depth/address contribution and retrains both stages; (ii) *formula only*, which removes both numerical encoders while preserving the fusion-head interface; (iii) *without correlation branch*,

Metric	MSE / MSE	MSE / TED
NMSE ↓	$(2.264 \pm 2.590)\text{e}{-3}$	$(2.702 \pm 1.109)\text{e}{-4}$
$\text{Acc}_{0.01}$ ↑ (%)	74.24 ± 10.74	89.39 ± 3.47
True TED ↓	0.361 ± 0.112	0.239 ± 0.010

Table 4: Matched Physics ablation: mean ± sample standard deviation across three seeds, with 44 tasks per seed. The numerical-only condition replaces TED by MSE throughout the structural-score path.

Verifier	Objective ↓	$\text{MAE}_{\leq 1}$ ↓	$\text{Recall}_{>1}$ ↑
w/o tree address	1.62e-1	1.79e-1	**52.63**
Formula only	2.29e-1	2.33e-1	36.84
w/o corr. branch	1.31e-1	1.35e-1	49.12
One-stage	1.47e-1	1.52e-1	35.09
Full verifier	**1.26e-1**	**1.31e-1**	**52.63**

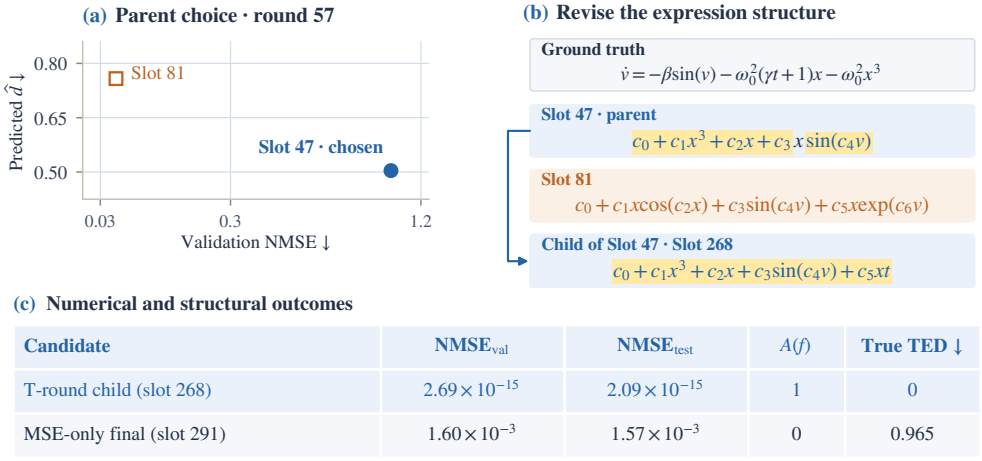
Table 5: Verifier ablations on the fixed synthetic validation set.

which removes the signed-correlation encoder; and (iv) *one-stage*, which retains the full architecture and inputs but optimizes the Stage 2 objective from random initialization. The formula-only and no-correlation variants start from the full model's Stage 1 checkpoint.

Table 5 shows that every ablation worsens both the objective and $\text{MAE}_{\leq 1}$ relative to the full verifier. Recall also decreases without numerical inputs, the correlation branch, or Stage 1, while the no-tree-address variant ties the full model. Together, these results indicate that each tested component is needed to attain the best overall verifier performance. Training details appear in Appendix E.6.

4.5 Case Study: Physics 11

In round 57 of Physics 11, a T-round selects a parent with worse validation NMSE (1.019 versus 0.0476) but lower predicted TED (0.504 versus 0.758). Its child separates sinusoidal damping and adds a time–position term, reaching test NMSE 2.09×10^{-15}. Figure 4 shows this choice. This example shows that the TED verifier can guide the LLM toward the correct formula.



(c) Numerical and structural outcomes

Candidate	NMSE_{val}	$\text{NMSE}_{\text{test}}$	$A(f)$	True TED ↓
T-round child (slot 268)	2.69×10^{-15}	2.09×10^{-15}	1	0
MSE-only final (slot 291)	1.60×10^{-3}	1.57×10^{-3}	0	0.965

Figure 4: PO11 case study. (a) Structural feedback selects slot 47 despite its higher validation NMSE. (b) Ground truth, candidate structures, and the child generated from slot 47; c_i denote fitted constants. (c) Numerical and structural outcomes.

5 Conclusion and Future Work

We introduced L2G-SR through three contributions: a modified TED for structural supervision, a verifier that estimates structural distance to an unseen target from observations and candidate formulas, and an LLM-guided search agent that combines structural and numerical feedback to select and refine formulas. Limited computational resources led us to evaluate lightweight, low-cost LLMs. Future work will optimize the framework with the latest LLMs. We will also explore alternative training strategies, including post-training LLMs as structural verifiers or directly post-training the analyst to improve formula revision.

AI USE STATEMENT

AI was used for writing code, debugging, running some experiments, polishing the manuscript, checking mathematical derivations, literature searches, and identifying relevant studies for parts of the related work.

REPRODUCIBILITY STATEMENT

Appendices A–D specify the structural target, verifier inputs, optimization settings, and search decisions. Appendix E records measurements, denominators, and the reporting protocol. The accompanying writing artifacts contain the source-linked result summaries and scripts used to construct the tables and figures.

ETHICS STATEMENT

The study concerns equation discovery from benchmark observations and synthetic mathematical expressions. A plausible expression or a small structural score is not a certificate of a physical law. The formulas discovered by this model do not necessarily represent actual physical laws, and users should assess their correctness and validity.

REFERENCES

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural Symbolic Regression that Scales. In *International Conference on Machine Learning*, pp. 936–945, 2021.

Bogdan Burlacu, Gabriel Kronberger, and Michael Kommenda. Operon C++: An Efficient Genetic Programming Framework for Symbolic Regression. In *Genetic and Evolutionary Computation Conference Companion*, pp. 1562–1570, 2020. doi: 10.1145/3377929.3398099.

Miles Cranmer. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl. arXiv:2305.01582, 2023.

Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. ISBN 9780521898065. doi: 10.1017/CBO9780511801655.

Arya Grayeli, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer, and Swarat Chaudhuri. Symbolic Regression with a Learned Concept Library. In *Advances in Neural Information Processing Systems*, volume 37, pp. 44678–44709. Curran Associates, Inc., 2024. doi: 10.52202/079017-1419. URL https://papers.nips.cc/paper_files/paper/2024/hash/4ec3ddc465c6d650c9c419fb91f1c00a-Abstract-Conference.html.

Samuel Holt, Zhaozhi Qian, and Mihaela van der Schaar. Deep Generative Symbolic Regression. In *International Conference on Learning Representations*, 2023.

Nan Jiang, Ziyi Wang, and Yexiang Xue. EGG-SR: Embedding Symbolic Equivalence into Symbolic Regression via Equality Graph. In *International Conference on Learning Representations*, pp. 13650–13689, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1704fe7aaff33a54802b83a016050ab8-Paper-Conference.pdf.

Ying Jin, Weilin Fu, Jian Kang, Jiadong Guo, and Jian Guo. Bayesian Symbolic Regression. arXiv:1910.08892, 2019.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end Symbolic Regression with Transformers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 10269–10281. Curran Associates, Inc., 2022. doi: 10.52202/068431-0746. URL https://papers.nips.cc/paper/2022/hash/42eb37cdbefd7abae0835f4b67548c39-Abstract-Conference.html.

Pierre-Alexandre Kamienny, Guillaume Lample, Sylvain Lamprier, and Marco Virgolin. Deep Generative Symbolic Regression with Monte-Carlo-Tree-Search. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 15655–15668, 2023.

John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. Contemporary Symbolic Regression Methods and Their Relative Performance. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P. Santiago, T. Nathan Mundhenk, Ignacio Aravena, Garrett Mulcahy, and Brenden K. Petersen. A Unified Framework for Deep Symbolic Regression. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-2463.

Yunlun Li and Sinno Jialin Pan. Robust Equation Structure Learning with Adaptive Refinement. In *International Conference on Learning Representations*, pp. 70029–70053, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/72416ded78a439907ff72165ac9c56e0-Paper-Conference.pdf.

Pingchuan Ma, Tsun-Hsuan Wang, Minghao Guo, Zhiqing Sun, Joshua B. Tenenbaum, Daniela Rus, Chuang Gan, and Wojciech Matusik. LLM and Simulation as Bilevel Optimizers: A New Paradigm to Advance Physical Scientific Discovery. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 33940–33962. PMLR, 2024. URL https://proceedings.mlr.press/v235/ma24m.html.

Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. Rethinking Symbolic Regression Datasets and Benchmarks for Scientific Discovery. *Journal of Data-centric Machine Learning Research*, 1(3):1–38, 2024. URL https://data.mlr.press/assets/pdf/v01-3.pdf.

Kazem Meidani, Parshin Shojaee, Chandan K. Reddy, and Amir Barati Farimani. SNIP: Bridging Mathematical Symbolic and Numeric Realms with Unified Pre-training. In *International Conference on Learning Representations*, pp. 42359–42372, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b935d7193defab655b5f5d5b6f07ad41-Abstract-Conference.html.

T. Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P. Santiago, Daniel M. Faissol, and Brenden K. Petersen. Symbolic Regression via Deep Reinforcement Learning Enhanced Genetic Programming Seeding. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Xinyu Pang, Zhanke Zhou, Xuan Li, Fangrui Lv, Shanshan Wei, Sen Cui, Bo Han, and Changshui Zhang. Deliberate Evolution: Agentic Reasoning for Sample-Efficient Symbolic Regression with LLMs. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/66263.

Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, and Joanne T. Kim. Deep Symbolic Regression: Recovering Mathematical Expressions from Data via Risk-Seeking Policy Gradients. In *International Conference on Learning Representations*, 2021.

Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu, Chenyang Wang, Chencheng Tang, Haoling Chang, Qi Liu, Ziheng Zhou, Tianyu Zhang, Jingtian Zhang, Zhangyi Liu, Minghao Li, Yuku Zhang, Boxuan Jing, Xianqi Yin, Yutong Ren, Zizhuo Fu, Jiaming Ji, Weike Wang, Xudong Tian, Anqi Lv, Laifu Man, Jianxiang Li, Feiyu Tao, Qihua Sun, Zhou Liang, Yushu Mu, Zhongxuan Li, Jing-Jun Zhang, Shutao Zhang, Xiaotian Li, Xingqi Xia, Jiawei Lin, Zheyu Shen, Jiahang Chen, Qiuhao Xiong, Binran Wang, Fengyuan Wang, Ziyang Ni, Bohan Zhang, Fan Cui, Changkun Shao, Qing-Hong Cao, Ming-xing Luo, Yaodong Yang, Muhan Zhang, and Hua Xing Zhu. PHYBench: Holistic Evaluation of Physical Perception and Reasoning in Large Language Models. In *Advances in*

Neural Information Processing Systems, volume 38, 2025. doi: 10.52202/085713-0044. URL https://papers.nips.cc/paper_files/paper/2025/hash/01e793f8cf5689aa0ea46a1b01071bea-Abstract-Datasets_and_Benchmarks_Track.html. EED code: https://github.com/phybench-official/phybench/tree/d9db3ec7246f3678aaee65d44a32649ad93beea2/EED.

Evgeny S. Saveliev, Samuel Holt, Nabeel Seedat, David L. Bentley, Jim Weatherall, and Mihaela van der Schaar. Influence-Guided Symbolic Regression: Scientific Discovery via LLM-Driven Equation Search with Granular Feedback. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/60987.

Alexander Scarlatos and Andrew Lan. Tree-Based Representation and Generation of Natural and Mathematical Language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3714–3730. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.205. URL https://aclanthology.org/2023.acl-long.205/.

Michael Schmidt and Hod Lipson. Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324(5923):81–85, 2009. doi: 10.1126/science.1165893.

Parshin Shojaee, Kazem Meidani, Amir Barati Farimani, and Chandan K. Reddy. Transformer-Based Planning for Symbolic Regression. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45907–45919, 2023. doi: 10.52202/075280-1990.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K. Reddy. LLM-SR: Scientific Equation Discovery via Programming with Large Language Models. In *International Conference on Learning Representations*, pp. 16054–16085, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/28df8e730c054c5331855fd4d5403ba9-Abstract-Conference.html.

Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D Doan, and Chandan K. Reddy. LLM-SRBench: A New Benchmark for Scientific Equation Discovery with Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55325–55359. PMLR, 2025b. URL https://proceedings.mlr.press/v267/shojaee25a.html.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A Physics-Inspired Method for Symbolic Regression. *Science Advances*, 6(16):eaay2631, 2020. doi: 10.1126/sciadv.aay2631.

Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI Feynman 2.0: Pareto-Optimal Symbolic Regression Exploiting Graph Modularity. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4860–4871, 2020.

Marco Virgolin and Solon P. Pissis. Symbolic Regression is NP-hard. *Transactions on Machine Learning Research*, 2022.

Marco Virgolin, Tanja Alderliesten, Cees Witteveen, and Peter A. N. Bosman. Improving Model-Based Genetic Programming for Symbolic Regression of Small Expressions. *Evolutionary Computation*, 29(2):211–237, 2021. doi: 10.1162/EVCO_A_00278.

Runxiang Wang, Boxiao Wang, Kai Li, Yifan Zhang, and Jian Cheng. DrSR: LLM based Scientific Equation Discovery with Dual Reasoning from Data and Experience. arXiv:2506.04282, 2025.

Shijie Xia, Yuhan Sun, and Pengfei Liu. SR-Scientist: Scientific Equation Discovery With Agentic AI. In *International Conference on Learning Representations*, pp. 75787–75811, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/7a9d48d13056b6b6a7f5bce1c3ba8f2d-Paper-Conference.pdf.

Chuyang Xiang, Yichen Wei, and Junchi Yan. MOD-SR: Unifying Multimodal Learning and Direct Optimization with Gradient-Guided Diffusion Model for Symbolic Regression. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/61549.

Zikai Xie, Wenmei Li, Man Luo, Jun Jiang, and Linjiang Chen. Language models guide symbolic equation discovery by controlling search. arXiv:2607.04156, 2026.

Zihan Yu, Jingtao Ding, Yong Li, and Depeng Jin. Symbolic regression via MDLformer-guided search: from minimizing prediction error to minimizing description length. In *International Conference on Learning Representations*, pp. 65306–65333, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a402493de088886740b5939f666a6e56-Abstract-Conference.html.

Kaizhong Zhang and Dennis Shasha. Simple Fast Algorithms for the Editing Distance between Trees and Related Problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989. doi: 10.1137/0218082.

Wenxiao Zhao, Dong Liu, Kaiyi Xu, Feng Liu, Zhen Zhao, Fei Ben, Shu Wang, Wenhao Li, Ying Nian Wu, Fenghua Ling, Haobo Li, and Lei Bai. A-SR: Self-Evolving Agentic LLMs for Symbolic Regression via Hierarchical Coordination. arXiv:2608.04872, 2026.

A STRUCTURAL TARGET: IMPLEMENTATION DETAILS

A.1 TREE CONSTRUCTION AND PREPROCESSING

Variables retain their identities. Numeric-only subexpressions become a single `number` leaf; symbolic constants recognized by the parser become `param`. Function names and operation types supply internal labels. Nested Add and Mul nodes are flattened recursively. An AST is the representation; TED is the subsequent comparison algorithm.

The three preprocessing views in Equation 4 are:

1. **Structure-preserving candidate view.** Parse the candidate without automatic evaluation, absorb parameter-only subexpressions, and collapse pure constants while preserving parse structure where possible. Process the reference using the ordinary evaluated view below.
2. **Ordinary evaluated view.** Use normal symbolic evaluation, absorb constant-only symbolic combinations, and collapse pure numeric subexpressions on both sides.
3. **Expanded view.** Absorb constants, expand products, expand power bases, combine powers, denest powers, collapse pure constants, and absorb remaining symbolic constant combinations.

The expanded view applies forced power transformations, and TED is evaluated after this preprocessing. Each view is normalized by its own reference-tree size before taking the minimum. The result is not clipped to $[0, 1]$.

A.2 COSTS AND LOCAL ASSIGNMENT

Node insertion and deletion cost one. Relabeling identical labels costs zero, as does exchanging `number` and `param`; other relabelings cost one. Whole-subtree insertion and deletion use Equation 2: for $s \leq 5$ the cost is s, and larger subtrees receive a discount. This extended ordered computation is evaluated for every root pair.

When both roots belong to $\{\mathrm{Add}, \mathrm{Mul}\}$, an additional child-assignment route is evaluated, including the root relabeling cost when necessary. Real-child matches recurse through the hybrid routine; padded matches pay a subtree insertion or deletion cost. The smaller ordered/assignment cost is retained. At other roots, the ordered routine is returned.

Changing a numeric exponent from two to three leaves the abstract power tree unchanged; changing $\sin$ to $\cos$ changes an operator label.

A.3 WORKED EXAMPLES ACROSS THE THREE VIEWS

We illustrate five distinct mechanisms using a candidate f and a reference g. Write D_k for the raw hybrid edit cost and $n_k = |T(P_k^r(g))|$ for the reference-tree size in view v_k. The normalized score is D_k/n_k, and the final TED is the minimum across the three views. Table 6 reports all three calculations. Ties are resolved in the order v_0, v_1, v_2. Figures 5 and 6 draw the actual candidate and reference trees after the selected preprocessing; N denotes a `number` leaf.

Table 6: Worked TED calculations. Each view reports the raw edit cost divided by its reference-tree size. The letters match the tree diagrams.

	Candidate f	Reference g	v_0	v_1	v_2	TED
(a)	$x + (y + z)$	$(x + y) + z$	$0/4$	$0/4$	$0/4$	0
(b)	$\sin(x + y)$	$\cos(x + y)$	$1/4$	$1/4$	$1/4$	$1/4$
(c)	$x(y + z)$	$xy + xz$	$5/7$	$5/7$	$0/7$	0
(d)	$(x + y)^2$	$x^2 + 2xy + y^2$	$9/11$	$9/11$	$0/11$	0
(e)	$x + \sin(yz + w)$	x	$6.6/1$	$6.6/1$	$6.6/1$	6.6

(a) Associativity. For $f = x + (y + z)$ and $g = (x + y) + z$, flattening nested Add nodes produces one Add root with leaves x, y, z on both sides. Each reference tree has four nodes, and no edits are needed: $v_0 = v_1 = v_2 = 0/4 = 0$. The selected view is v_0.

(b) Function relabeling. For $f = \sin(x + y)$ and $g = \cos(x + y)$, the Add subtree and its variable leaves agree. Only the root function label changes, costing one. All three reference trees have four nodes, so $v_0 = v_1 = v_2 = 1/4$; v_0 is selected. The highlighted roots show this single edit.

(c) Product expansion. For $f = x(y + z)$ and $g = xy + xz$, views v_0 and v_1 retain a Mul root on the candidate and an Add root on the reference. A minimum-cost assignment changes the root at cost one, matches x to xy at cost two (insert Mul and y), and matches $y + z$ to xz at cost two (Add to Mul and y to x). Thus $D_0 = D_1 = 1 + 2 + 2 = 5$ and $n_0 = n_1 = 7$. View v_2 distributes the product, making both trees identical with seven nodes: $v_2 = 0/7$. The selected score is zero.

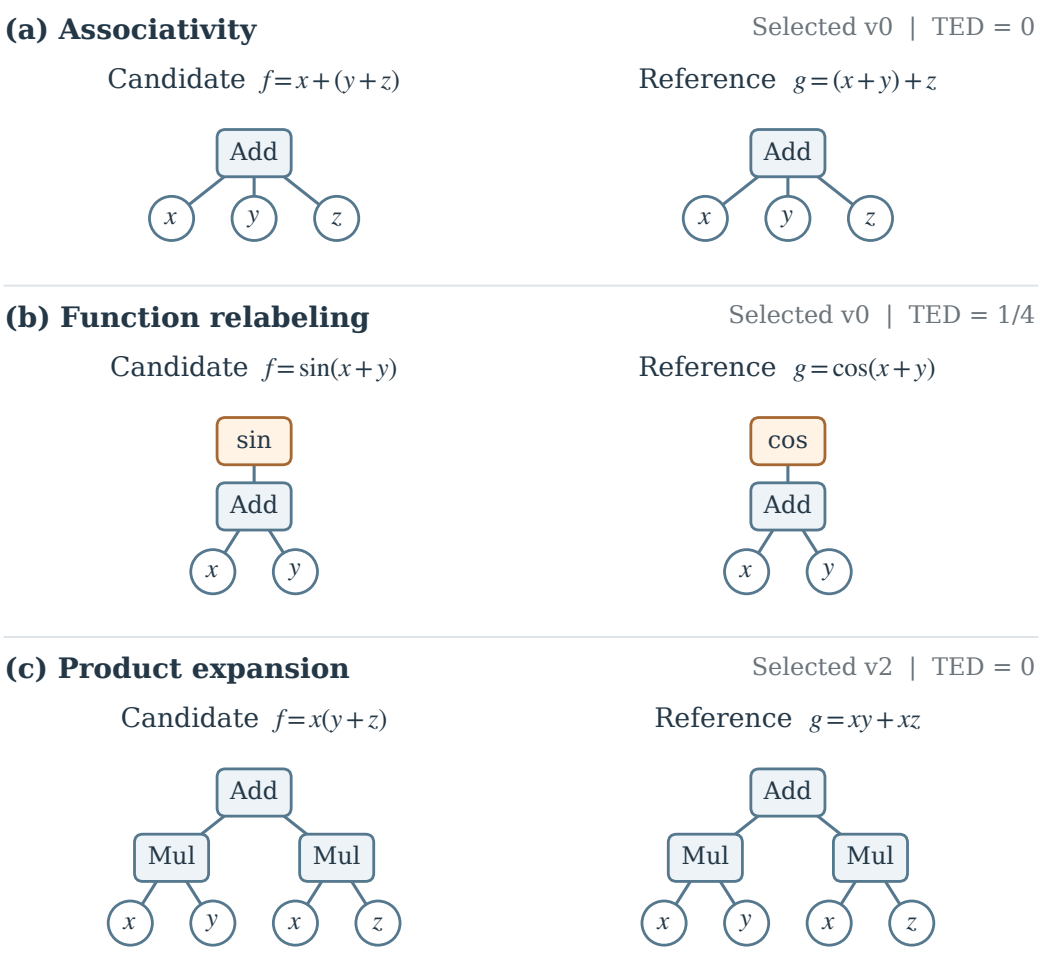


Figure 5: Selected abstract trees for examples (a)–(c). Input formulas appear above each tree; the diagrams show the trees after the indicated preprocessing. Flattening removes grouping differences in (a), the highlighted function labels require one edit in (b), and product expansion gives identical trees in (c).

(d) Polynomial expansion. For $f = (x + y)^2$ and $g = x^2 + 2xy + y^2$, views v_0 and v_1 compare $\mathrm{Pow}(\mathrm{Add}(x, y), N)$ with an Add root whose children are $\mathrm{Pow}(x, N)$, $\mathrm{Pow}(y, N)$, and $\mathrm{Mul}(N, x, y)$. An optimal ordered edit changes the root at cost one; converts $\mathrm{Add}(x, y)$ to $\mathrm{Pow}(x, N)$ at cost two; converts the remaining N to $\mathrm{Pow}(y, N)$ at cost two; and inserts the four-node mixed term at cost four. Hence $D_0 = D_1 = 1 + 2 + 2 + 4 = 9$, with $n_0 = n_1 = 11$. View v_2 expands the square, giving identical 11-node trees and the selected score $0/11 = 0$. Numeric exponent and coefficient leaves are all labeled N.

(e) Subtree deletion discount. For $f = x+\sin(yz+w)$ and $g = x$, all views retain the extra branch. The outlined branch has six nodes: sin, Add, Mul, y, z, w. Deleting it costs $\kappa(6) = 0.6(6-5)+5 =$

5.6; deleting the outer Add root costs one more and leaves x. Therefore $D_0 = D_1 = D_2 = 6.6$, while every reference tree contains only one node. All three normalized scores are $6.6/1 = 6.6$, and v_0 is selected. This example also shows why normalization by the reference size does not bound the score by one.

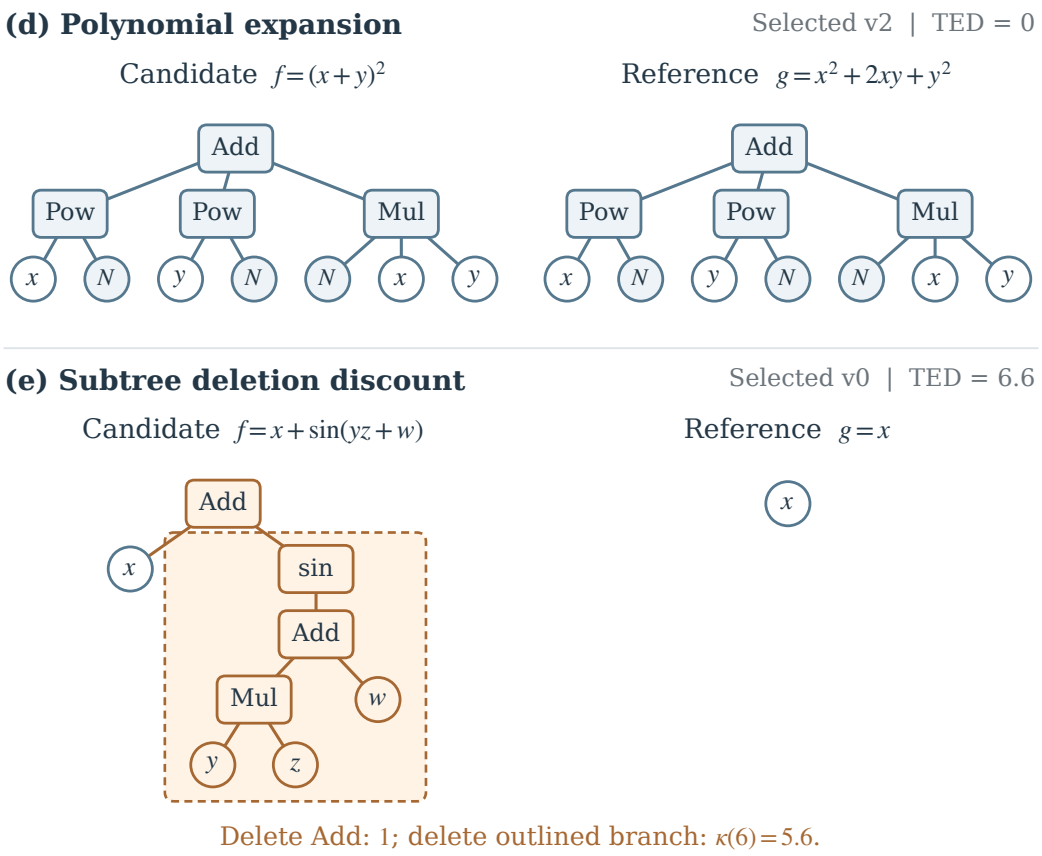


Figure 6: Selected abstract trees for examples (d)–(e). In (d), expanding the square yields identical trees; N denotes an abstract numeric leaf. In (e), the dashed outline contains the six-node branch deleted at discounted cost 5.6; the highlighted Add root incurs another unit deletion cost. Only the variable x is retained.

A.4 Additional examples of partial structural matches

Table 7 gives six further formula pairs, continuing the notation above. All select v_0. The tree diagrams preserve the child order returned by preprocessing; commutative assignment can still match children across different positions. Orange nodes indicate deletions, insertions, or relabelings along one minimum-cost edit path. As before, N denotes a `number` leaf.

(f) Extra multiplicative factor. For $f = x^2 \sin y$ and $g = \sin y$, delete the three-node factor $\mathrm{Pow}(x, N)$ and the outer Mul root, retaining the entire $\sin y$ subtree. The cost is $3 + 1 = 4$, and the reference has two nodes. All views therefore give $4/2 = 2$.

(g) Product versus sum. For $f = (x + y)(x + z)$ and $g = xy + yz$, the unexpanded view changes the Mul root to Add at cost one, changes $\mathrm{Add}(x, y)$ to $\mathrm{Mul}(x, y)$ at cost one, and changes $\mathrm{Add}(x, z)$ to $\mathrm{Mul}(y, z)$ at cost two. Thus $v_0 = v_1 = (1+1+2)/7 = 4/7$. Expansion gives $x^2+xy+xz+yz$; deleting the extra three-node terms x^2 and xz gives $v_2 = 6/7$. The unexpanded view has the smaller distance.

(h) Cube versus separate cubes. For $f = (x + y)^3$ and $g = x^3 + y^3$, v_0 and v_1 compare $\mathrm{Pow}(\mathrm{Add}(x, y), N)$ with $\mathrm{Add}(\mathrm{Pow}(x, N), \mathrm{Pow}(y, N))$. Change the root at cost one, convert

Table 7: Additional worked TED calculations. Each view reports the raw edit cost divided by its reference-tree size. All six pairs select v_0, using the first view in a tie.

	Candidate f	Reference g	v_0	v_1	v_2	TED
(f)	$x^2 \sin y$	$\sin y$	$4/2$	$4/2$	$4/2$	2
(g)	$(x+y)(x+z)$	$xy+yz$	$4/7$	$4/7$	$6/7$	$4/7$
(h)	$(x+y)^3$	x^3+y^3	$5/7$	$5/7$	$11.2/7$	$5/7$
(i)	$\sqrt{x+y}$	$\sqrt{x}$	$2/3$	$2/3$	$2/3$	$2/3$
(j)	$(x+y)(x+z)$	$x(x+y)$	$2/5$	$2/5$	$6/7$	$2/5$
(k)	$(x+y)^2$	$(x+y)^3$	$0/5$	$0/5$	$7.6/19$	0

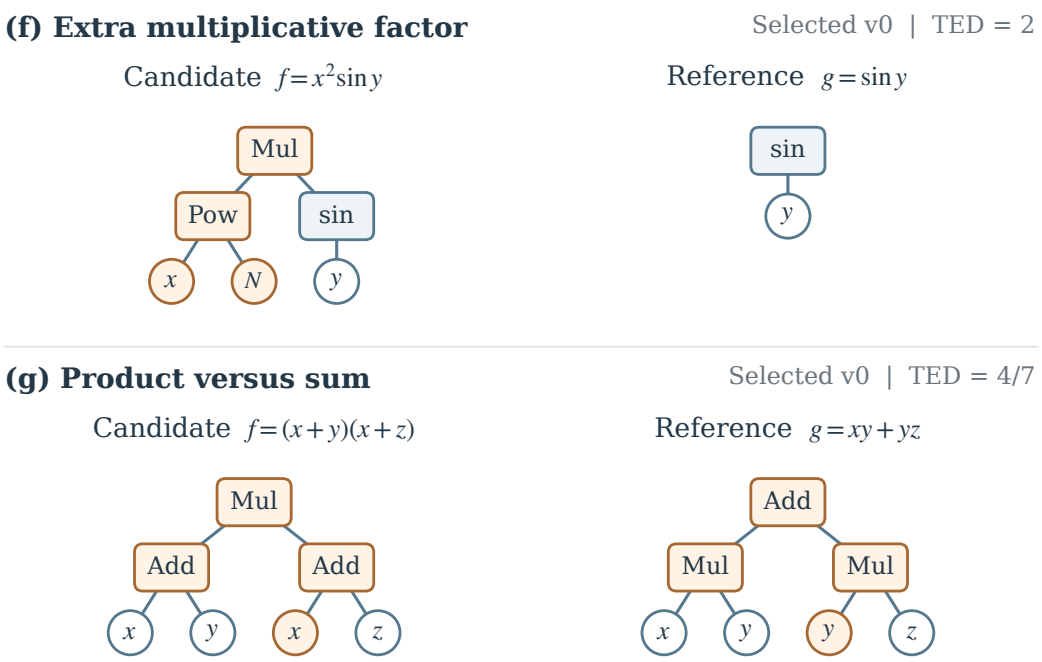


Figure 7: Selected v_0 trees for (f)–(g). Orange marks nodes edited along a minimum-cost path. In (f), the extra power subtree and Mul root are deleted. In (g), four relabelings transform the product of sums into the reference sum of products.

$\mathrm{Add}(x, y)$ to $\mathrm{Pow}(x, N)$ at cost two, and turn the remaining N into $\mathrm{Pow}(y, N)$ at cost two. Hence $v_0 = v_1 = 5/7$. After expansion, the two additional mixed terms $3x^2y$ and $3xy^2$ each have six nodes. Their discounted deletion costs give $v_2 = 2\kappa(6)/7 = 11.2/7 = 1.6$, so the final score is $5/7$.

(i) Square-root argument. For $f = \sqrt{x+y}$ and $g = \sqrt{x}$, the trees are $\mathrm{Pow}(\mathrm{Add}(x, y), N)$ and $\mathrm{Pow}(x, N)$. The exponent $1/2$ becomes one numeric leaf. Retain the Pow root, x, and N; delete the inner Add and y. All views give $2/3$.

(j) Shared additive factor. For $f = (x + y)(x + z)$ and $g = x(x + y)$, commutative assignment matches the common $\mathrm{Add}(x, y)$ subtree across child positions. In the other factor, delete Add and z from $x + z$, retaining x. The raw cost is two and the reference has five nodes, giving $v_0 = v_1 = 2/5$. Expansion produces $x^2 + xy + xz + yz$ and $x^2 + xy$; deleting the two extra three-node products gives $v_2 = 6/7$. The final score is $2/5$.

(k) Numeric exponent abstraction. For $f = (x + y)^2$ and $g = (x + y)^3$, v_0 and v_1 both produce $\mathrm{Pow}(\mathrm{Add}(x, y), N)$, so their scores are $0/5$. In v_2, the pure-power terms still match because numeric exponents are abstracted. Matching the candidate mixed term $2xy$ to one reference mixed term requires two insertions; inserting the remaining six-node mixed term costs $\kappa(6) = 5.6$. The

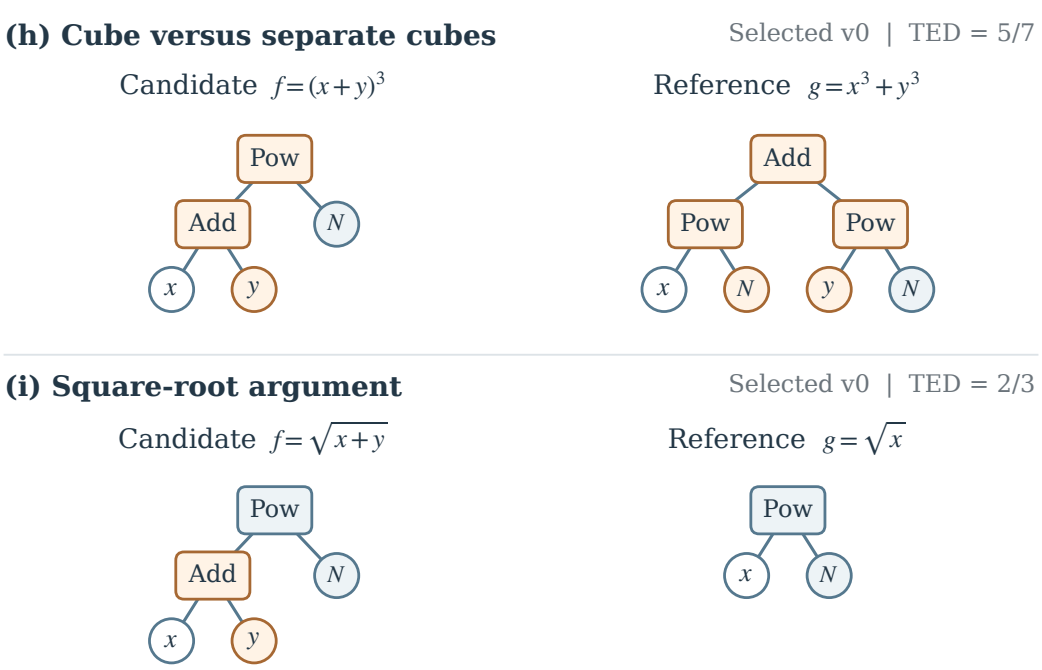


Figure 8: Selected v_0 trees for (h)–(i). In (h), orange marks three relabelings and two inserted nodes; the original exponent leaf is retained as the reference's rightmost N. In (i), deleting the orange Add and y nodes leaves the reference square-root tree. Numeric exponents are represented by N.

reference has 19 nodes, giving $v_2 = (2 + 5.6)/19 = 0.4$. The final TED is zero, illustrating that zero structural distance does not imply algebraic equivalence.

B Synthetic formula and observation generation

We describe the archived August 30 configuration. Listed probabilities are rounded proposals; feasibility checks and training resampling alter the resulting distributions. Categorical weights are renormalized over permitted outcomes.

B.1 Empirical priors and formula size

1. **Source formulas and weighting.** Priors use 32,177 LLM-SR formulas from 153 problems, 20,400 PySR formulas from 239 problems, and 120 folded SRBench/Feynman targets, each treated as one problem. Variable, root, operator, and constant statistics give equal weight to sources, problems within each source, and formulas within each problem:
$$w(g, p, s) = \frac{1}{3|\mathcal{P}_s||\mathcal{F}_{s,p}|}, \tag{12}$$
Here $\mathcal{P}_s$ contains source s's problems, and $\mathcal{F}_{s,p}$ contains its formulas for problem p. Before counting, negative powers use inverses, and exponents $1/2$, 2, and 3 use square-root, square, and cube operators.
2. **Node-count and operator-count prior.** Let N count all nodes, B binary operators (e.g., addition or division), and U unary operators (e.g., sine or logarithm). With $B + 1$ leaves,
$$N = 2B + U + 1.$$
For this joint size prior only, the source weights are $(\pi_{\text{GT}}, \pi_{\text{PySR}}, \pi_{\text{LLM-SR}}) = (0.2, 0.2, 0.6)$:
$$P_0(N = n, B = b) = \sum_{s,p,g} \frac{\pi_s}{|\mathcal{P}_s||\mathcal{F}_{s,p}|} \mathbf{1}[N(g) = n,\ B(g) = b]. \tag{13}$$

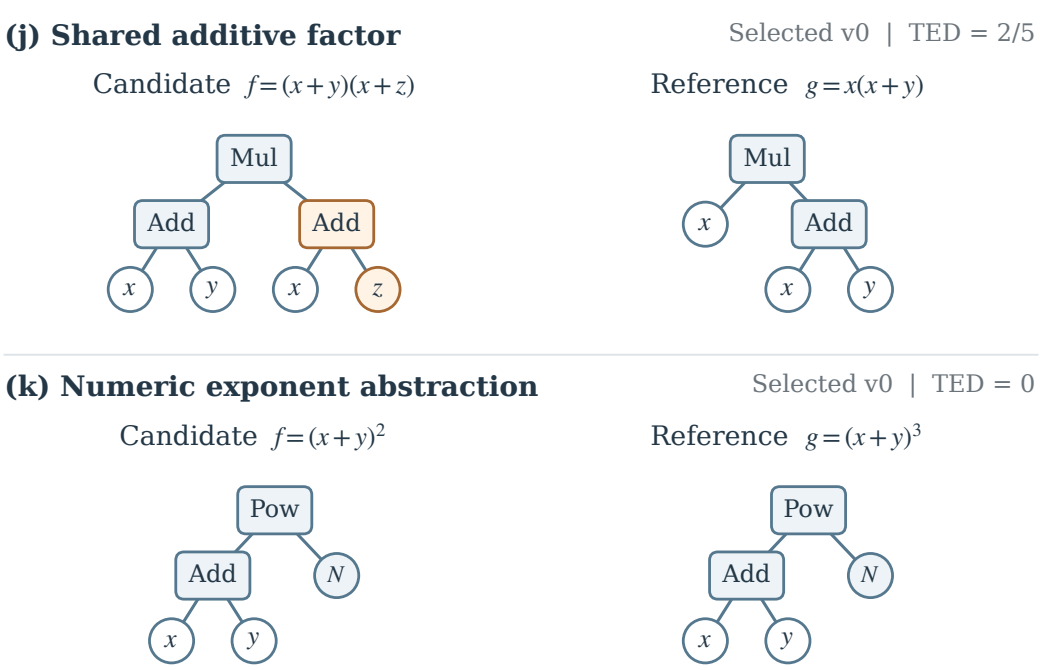


Figure 9: Selected v_0 trees for (j)–(k). In (j), commutative matching preserves the common additive factor; deleting the orange Add and z nodes supplies the remaining x factor. In (k), both exponent values become the same numeric leaf N, making the abstract trees identical.

Exclude formulas exceeding 100 nodes, then adjust the length marginal while preserving $P_0(B \mid N)$. Draw integer N from the adjusted row sums and B from that row's feasible entries. Table 8 groups lengths for display only. The legacy length-bin floor of 0.001 is inactive.

3. **Variable counts, root families, and feasibility.** Limit formulas to 100 nodes and 10 variables. Clip variable-bin upper endpoints to $\min(10, \lfloor (N+1)/2 \rfloor)$, remove empty bins, then select a bin and draw an integer uniformly within it. A one-node tree is a variable; $N = 2$ forces a unary root; $N > 2$ excludes a leaf root. The $B + 1$ leaves must accommodate all variables after reserving exponent leaves. A unary root or required inverse needs a unary operator; a non-division binary root with required division needs $B \geq 2$. Retry infeasible proposals.

B.1.1 Theory of mixture priors and conditional sampling

Let P_s be the empirical law obtained by averaging formulas within each problem and then problems within source s, as specified above. The mixture is $P_\pi = \sum_s \pi_s P_s$. For the joint size prior, $\pi = (0.2, 0.2, 0.6)$ in GT/PySR/LLM-SR order, and each source contains only formulas retained after the 100-node cutoff and their nonempty problems. Operator statistics instead use $\pi_s = 1/3$.

Proposition 1 (Conditional sampling from the mixed size prior)**.** *For any sampled length n with $P_\pi(N = n) > 0$, the source weights conditional on that length and the resulting binary-count law are*

$$\begin{aligned} \alpha_s(n) &= \frac{\pi_s P_s(N = n)}{\sum_t \pi_t P_t(N = n)}, \\ P_0(B = b \mid N = n) &= \frac{\sum_s \pi_s P_s(N = n, B = b)}{\sum_s \pi_s P_s(N = n)} = \sum_s \alpha_s(n) P_s(B = b \mid N = n). \end{aligned} \tag{14}$$

Sources with no mass at n are omitted. Thus normalizing the mixed joint row implements a source draw from $\alpha(n)$ followed by a source-specific conditional count draw.

Proof. Let S be the selected source and $Z_n = \sum_s \pi_s P_s(N = n) > 0$. The joint and marginal masses are

$$P_\pi(S = s, N = n, B = b) = \pi_s P_s(N = n, B = b),$$
$$P_\pi(S = s, N = n) = \sum_b \pi_s P_s(N = n, B = b) = \pi_s P_s(N = n),$$
$$P_\pi(N = n) = \sum_s P_\pi(S = s, N = n) = Z_n.$$

Hence

$$\alpha_s(n) = \frac{P_\pi(S = s, N = n)}{P_\pi(N = n)} = \frac{\pi_s P_s(N = n)}{Z_n} \geq 0, \qquad \sum_s \alpha_s(n) = \frac{Z_n}{Z_n} = 1.$$

A zero-mass source satisfies $P_s(N = n) = 0 \Rightarrow P_s(N = n, B = b) = 0$ for every b. On the remaining sources,

$$\begin{aligned} P_0(B = b \mid N = n) &= \frac{\sum_s \pi_s P_s(N = n, B = b)}{Z_n} \\ &= \sum_{s:P_s(N=n)>0} \frac{\pi_s P_s(N = n)}{Z_n} P_s(B = b \mid N = n) \\ &= \sum_{s:P_s(N=n)>0} \alpha_s(n) P_s(B = b \mid N = n). \end{aligned}$$

The resulting law is normalized and equals the two-stage sampling law:

$$\sum_b P_0(B = b \mid N = n) = \sum_{s:P_s(N=n)>0} \alpha_s(n) \underbrace{\sum_b P_s(B = b \mid N = n)}_{1} = 1,$$

$$\begin{aligned} \Pr_{\text{two-stage}}(B = b \mid n) &= \sum_{s:P_s(N=n)>0} \Pr(S = s \mid n) \Pr(B = b \mid S = s, n) \\ &= \sum_{s:P_s(N=n)>0} \alpha_s(n) P_s(B = b \mid N = n) = P_0(B = b \mid N = n). \end{aligned}$$

□

This justifies mixing joint counts before normalization: conditioning on length changes the appropriate source weights from π_s to $\alpha_s(n)$.

Proposition 2 (Minimal-change length reweighting and feasible sampling)**.** *Let $\bar{P}_0$ be the normalized retained joint size law, $p_n = \bar{P}_0(N = n)$, and q the adjusted length distribution, supported where $p_n > 0$. Row scaling gives*

$$Q(n, b) = \frac{q_n}{p_n} \bar{P}_0(n, b) = q_n \bar{P}_0(b \mid n), \tag{15}$$

with zero mass on unused rows. This law has length marginal q and preserves the binary-count conditional at each used length. Among joint laws $R(N, B)$ supported on $\bar{P}_0$ with length marginal q, it uniquely minimizes $D_{\mathrm{KL}}(R\|\bar{P}_0)$, attaining the minimum $D_{\mathrm{KL}}(q\|p)$.

Let h collect the sampled variable count, root family, and division/inverse requirements, and let $\mathcal{B}(n, h)$ be the admitted binary counts. For $q_n > 0$ and positive admissible row mass, restricting either $Q(n, \cdot)$ or $\bar{P}_0(n, \cdot)$ gives the same sampling kernel:

$$K_B(b \mid n, h) = \frac{\bar{P}_0(n, b)\, \mathbf{1}[b \in \mathcal{B}(n, h)]}{\sum_{b' \in \mathcal{B}(n,h)} \bar{P}_0(n, b')}. \tag{16}$$

Proof. **Marginal and conditional laws.** For $q_n > 0$, the support assumption gives $p_n > 0$, and

$$\sum_b Q(n,b) = \frac{q_n}{p_n}\sum_b \bar{P}_0(n,b) = \frac{q_n}{p_n}p_n = q_n,$$
$$\sum_{n,b} Q(n,b) = \sum_n q_n = 1,$$
$$Q(b \mid n) = \frac{Q(n,b)}{q_n} = \frac{\bar{P}_0(n,b)}{p_n} = \bar{P}_0(b \mid n),$$
$$\Pr_{\text{sampler}}(N = n, B = b) = q_n \bar{P}_0(b \mid n) = Q(n,b).$$

Rows with $q_n = 0$ have $Q(n,b) = 0$ for all b.

KL decomposition. For an admissible R, set $r_n(b) = R(b \mid n)$ and $t_n(b) = \bar{P}_0(b \mid n)$ on rows with $q_n > 0$. Then

$$R(n,b) = q_n r_n(b), \qquad \bar{P}_0(n,b) = p_n t_n(b), \qquad \sum_b r_n(b) = \sum_b t_n(b) = 1.$$

In KL sums, zero-probability terms are interpreted as zero, using $0 \log 0 = 0$. Since R is supported on $\bar{P}_0$, every denominator at a positive-mass term is positive. Expanding the logarithm gives

$$\begin{aligned} D_{\mathrm{KL}}(R\|\bar{P}_0) &= \sum_{n:q_n>0}\sum_b q_n r_n(b) \log \frac{q_n r_n(b)}{p_n t_n(b)} \\ &= \sum_{n:q_n>0} q_n \sum_b r_n(b)\left(\log\frac{q_n}{p_n} + \log\frac{r_n(b)}{t_n(b)}\right) \\ &= \sum_{n:q_n>0} q_n \log\frac{q_n}{p_n}\underbrace{\sum_b r_n(b)}_{1} + \sum_{n:q_n>0} q_n \sum_b r_n(b)\log\frac{r_n(b)}{t_n(b)} \\ &= D_{\mathrm{KL}}(q\|p) + \sum_{n:q_n>0} q_n D_{\mathrm{KL}}(r_n\|t_n). \end{aligned} \tag{17}$$

Optimality and uniqueness. The standard inequality $x \log x - x + 1 \geq 0$ for $x \geq 0$, with equality only at $x = 1$, yields

$$D_{\mathrm{KL}}(r_n\|t_n) = \sum_{b:t_n(b)>0} t_n(b)\left[\frac{r_n(b)}{t_n(b)}\log\frac{r_n(b)}{t_n(b)} - \frac{r_n(b)}{t_n(b)} + 1\right] \geq 0,$$
$$D_{\mathrm{KL}}(r_n\|t_n) = 0 \quad \Longleftrightarrow \quad r_n = t_n.$$

The inserted linear terms sum to $-\sum_b r_n(b) + \sum_b t_n(b) = 0$. Therefore

$$D_{\mathrm{KL}}(R\|\bar{P}_0) \geq D_{\mathrm{KL}}(q\|p) = D_{\mathrm{KL}}(Q\|\bar{P}_0),$$
$$\begin{aligned} D_{\mathrm{KL}}(R\|\bar{P}_0) = D_{\mathrm{KL}}(q\|p) &\Longleftrightarrow r_n = t_n \text{ for every } n \text{ with } q_n > 0 \\ &\Longleftrightarrow R(n,b) = q_n\bar{P}_0(b \mid n) = Q(n,b) \text{ on all used rows} \\ &\Longleftrightarrow R = Q. \end{aligned}$$

The last equivalence also uses $q_n = 0 \Rightarrow R(n,b) = Q(n,b) = 0$ for all b.

Feasibility conditioning. At fixed (n,h) with $q_n > 0$, write $\mathcal{B} = \mathcal{B}(n,h)$ and $A_{n,h} = \sum_{b' \in \mathcal{B}} \bar{P}_0(n,b') > 0$. Then

$$\begin{aligned} K_B(b \mid n,h) &= \frac{Q(b \mid n)\mathbf{1}[b \in \mathcal{B}]}{\sum_{b' \in \mathcal{B}} Q(b' \mid n)} \\ &= \frac{Q(n,b)\mathbf{1}[b \in \mathcal{B}]}{\sum_{b' \in \mathcal{B}} Q(n,b')} \\ &= \frac{(q_n/p_n)\bar{P}_0(n,b)\mathbf{1}[b \in \mathcal{B}]}{(q_n/p_n)A_{n,h}} = \frac{\bar{P}_0(n,b)\mathbf{1}[b \in \mathcal{B}]}{A_{n,h}}, \end{aligned}$$
$$\sum_b K_B(b \mid n,h) = \frac{A_{n,h}}{A_{n,h}} = 1.$$

Thus feasibility conditioning cancels both the length marginal and the row multiplier, giving Equation 16. □

The KL characterization makes row scaling a minimal-change update of the empirical joint law under the prescribed length marginal; the sampler realizes this minimizer directly. A zero denominator in the feasible kernel triggers a retry. This result concerns the size proposal and its local feasibility mask: subsequent operator assignment, exponent-leaf reservation, and formula filtering add further constraints. Structural retries may redraw N, and simplification may change its value, so the final accepted length distribution need not equal q. The separate empirical priors and sequential masks do not define a single empirically learned joint law over all formula properties.

Proposition 3 (Conditional operator priors from weighted occurrences)**.** *For operator o of arity a, let $m_s(o) = \mathbb{E}_s[c_o(G)]$ and $m_s(a) = \mathbb{E}_s[c_a(G)]$ under the same problem/formula hierarchy. If $\sum_s \pi_s m_s(a) > 0$, mixing occurrence counts and conditioning on arity gives*

$$P(o \mid a) = \frac{\sum_s \pi_s m_s(o)}{\sum_s \pi_s m_s(a)} = \sum_{s:m_s(a)>0} \underbrace{\frac{\pi_s m_s(a)}{\sum_t \pi_t m_t(a)}}_{\alpha_s(a)} \frac{m_s(o)}{m_s(a)}. \tag{18}$$

Here $\alpha_s(a)$ is the source probability conditional on an arity-a occurrence.

Proof. Let $\mathcal{O}_a$ be the operators of arity a and $\mu(s,o) = \pi_s m_s(o)$ their weighted occurrence mass. Since $c_a(G) = \sum_{o\in\mathcal{O}_a} c_o(G)$,

$$m_s(a) = \sum_{o\in\mathcal{O}_a} m_s(o), \qquad M_a := \sum_s \sum_{o\in\mathcal{O}_a} \mu(s,o) = \sum_s \pi_s m_s(a) > 0.$$

For the occurrence source S, label O, and arity A,

$$P(S=s, O=o \mid A=a) = \frac{\mu(s,o)}{M_a} = \frac{\pi_s m_s(o)}{M_a}, \qquad o \in \mathcal{O}_a,$$

$$\sum_s \sum_{o\in\mathcal{O}_a} P(S=s, O=o \mid A=a) = \frac{M_a}{M_a} = 1,$$

$$P(S=s \mid A=a) = \sum_{o\in\mathcal{O}_a} \frac{\pi_s m_s(o)}{M_a} = \frac{\pi_s m_s(a)}{M_a} = \alpha_s(a).$$

For $\pi_s m_s(a) > 0$, conditioning within the source gives

$$P(O=o \mid S=s, A=a) = \frac{P(S=s, O=o \mid A=a)}{P(S=s \mid A=a)} = \frac{\pi_s m_s(o)/M_a}{\pi_s m_s(a)/M_a} = \frac{m_s(o)}{m_s(a)}.$$

Consequently,

$$P(O=o \mid A=a) = \sum_{s:m_s(a)>0} \alpha_s(a) \frac{m_s(o)}{m_s(a)}$$

$$= \sum_{s:m_s(a)>0} \frac{\pi_s m_s(a)}{M_a} \frac{m_s(o)}{m_s(a)} = \frac{\sum_s \pi_s m_s(o)}{M_a},$$

$$\sum_{o\in\mathcal{O}_a} P(O=o \mid A=a) = \frac{\sum_s \pi_s \sum_{o\in\mathcal{O}_a} m_s(o)}{M_a} = \frac{M_a}{M_a} = 1.$$

Here $m_s(a) = 0 \Rightarrow m_s(o) = 0$ for every $o \in \mathcal{O}_a$; sources of zero weighted mass contribute zero and require no conditional draw. □

Even with equal source weights $\pi_s = 1/3$, the conditional source contribution depends on its weighted arity count. Thus the implementation mixes occurrence counts before dividing, rather than averaging per-formula operator ratios. The unary floor and contextual exclusions are subsequent modifications of this base law.

Table 8: Effective size, variable-count, and root-family proposals before feasibility constraints. The variable and root columns are separate categorical distributions.

Quantity	Outcome	Probability	Quantity	Outcome	Probability
N	1–2	0.001000000	Variables	1	0.127767302
	3–7	0.030000000		2	0.352240995
	8–15	0.321712885		3	0.282177813
	16–30	0.440506600		4–5	0.196282613
	31–50	0.184595015		6–8	0.041531276
	51–70	0.018782916		9–10	0
	71–85	0.002822544	Root	additive	0.624594682
	86–100	0.000580040		multiplicative	0.101696355
				rational	0.233553641
				power	0.000989314
				unary composition	0.038926974
				leaf	0.000239034

B.2 Tree shape and operator assignment

1. **Ordered tree shape.** For B binary operators, there are $C_B = \binom{2B}{B}/(B+1)$ ordered skeletons. Recursively choose left size i and right size $B-1-i$ with probability $C_i C_{B-1-i}/C_B$, giving each skeleton probability $1/C_B$.
2. **Placement of the remaining operators.** Across $2B+1$ skeleton positions, distribute $U = N - 2B - 1$ unary operators using a uniform weak composition $(u_1, \ldots, u_{2B+1})$ summing to U; wrap position j in u_j operators. A unary root reserves one operator before distributing $U-1$; a binary root receives zero, with U distributed elsewhere.
3. **Operator probabilities.** For an operator o of arity a, the empirical weight is
$$P(o \mid a) = \frac{\sum_{s,p,g} w(g,p,s) c_o(g)}{\sum_{s,p,g} w(g,p,s) c_a(g)},$$
Here c_o counts operator o and c_a counts all arity-a operators. Floor rare unary probabilities at 0.001 and proportionally rescale the rest (Table 9).
4. **Root choices and division/inverse requirements.** An additive root chooses `Add`/`Sub` with probabilities $(0.656837367, 0.343162633)$; multiplicative, rational, and power roots force `Mul`, `Div`, and `Pow`, respectively. The formula-level division proposal is 0.592371744 and the inverse proposal is 0.024717228. A rational root already requires division; otherwise, the probability of requiring it is
$$\frac{0.592371744 - 0.233553641}{1 - 0.233553641} = 0.468158142.$$
For $N = 1$, disable both requirements. A unary root is `Inv` if required; otherwise exclude `Inv`. Force any still-required `Div`/`Inv` at a uniformly chosen available position; exclude unrequired ones from unassigned nodes. Multiply the `Pow` weight by 0.15 for nonleaf exponents. Renormalize remaining weights and apply the checks in Appendix B.4.

B.2.1 Theory of the Catalan shape sampler

Condition on the sampled counts $B = b$ and $U = u$, so that $N = 2b + u + 1$. We analyze ordered, unlabeled tree shapes; left and right child positions remain distinct even when their eventual operator is commutative.

We use the standard Catalan count $C_b = \binom{2b}{b}/(b+1)$ for ordered full binary skeletons, with $C_0 = 1$ and $C_b = \sum_{i=0}^{b-1} C_i C_{b-1-i}$ for $b \geq 1$ (Flajolet & Sedgewick, 2009, pp. 6–7).

Proposition 4 (Uniform sampling of ordered binary skeletons)**.** *At each binary node with a remaining budget $b \geq 1$, draw the left-subtree budget I with*
$$P(I = i \mid b) = \frac{C_i C_{b-1-i}}{C_b}, \qquad i = 0, \ldots, b-1, \tag{19}$$

then draw both subtrees independently by the same rule. Every ordered full binary skeleton with b binary nodes has probability $1/C_b$.

Proof. **Valid split probabilities.** The stated Catalan recurrence gives

$$P(I = i \mid b) = \frac{C_i C_{b-1-i}}{C_b} \geq 0, \qquad \sum_{i=0}^{b-1} P(I = i \mid b) = \frac{\sum_{i=0}^{b-1} C_i C_{b-1-i}}{C_b} = 1.$$

Each recursive call has fewer binary nodes:

$$0 \leq I \leq b-1, \qquad 0 \leq b-1-I \leq b-1,$$

so recursion terminates at the unique $b = 0$ leaf.

Uniformity by induction. For $b = 0$, $P(\text{leaf} \mid 0) = 1 = 1/C_0$. Assume

$$P(T' \mid k) = \frac{1}{C_k} \qquad \text{for every } 0 \leq k < b \text{ and every skeleton } T' \text{ of size } k.$$

A skeleton T of size b uniquely determines its ordered subtrees and split:

$$T \longleftrightarrow (i, T_L, T_R), \qquad |T_L|_{\text{binary}} = i, \quad |T_R|_{\text{binary}} = j = b-1-i.$$

Conditional independence of the recursive draws and the induction hypothesis imply

$$\begin{aligned} P(T \mid b) &= P(I = i \mid b) P(T_L, T_R \mid I = i, b) \\ &= P(I = i \mid b) P(T_L \mid i) P(T_R \mid j) \\ &= \frac{C_i C_j}{C_b} \frac{1}{C_i} \frac{1}{C_j} = \frac{1}{C_b}. \end{aligned}$$

There is no second construction path for the same ordered tree. For the set $\mathcal{S}_b$ of all such skeletons,

$$\sum_{T \in \mathcal{S}_b} P(T \mid b) = |\mathcal{S}_b| \frac{1}{C_b} = 1.$$

□

Proposition 5 (Uniform unary–binary shapes with root constraints)**.** *Fix $b, u \geq 0$ and index the $K = 2b + 1$ skeleton positions in preorder, with position one at the root. Independently draw a uniform skeleton and a uniform nonnegative integer vector $(u_1, \ldots, u_K)$ summing to u, then insert u_j unary nodes above position j. Every ordered shape with these arity counts has probability*

$$P(T \mid b, u) = \frac{1}{C_b W(u, K)}, \qquad W(u, K) = \binom{u + K - 1}{K - 1}. \tag{20}$$

A prescribed unary root restricts the vector to $u_1 \geq 1$; a prescribed binary root requires $u_1 = 0$ and $b \geq 1$. The respective counts are

$$W_{\text{unary}} = \binom{u + K - 2}{K - 1} \quad (u \geq 1), \qquad W_{\text{binary}} = \binom{u + K - 2}{K - 2} \quad (b \geq 1). \tag{21}$$

A uniform draw among the admissible vectors assigns probability $1/(C_b W_r)$ to each shape with root arity r.

Proof. **One-to-one shape construction.** Write

$$\mathcal{U}_{u,K} = \left\{ (u_1, \ldots, u_K) \in \mathbb{Z}_{\geq 0}^K : \sum_{j=1}^{K} u_j = u \right\},$$

and let $\mathcal{T}_{b,u}$ be the target ordered shapes. Insertion defines

$$\Phi : \mathcal{S}_b \times \mathcal{U}_{u,K} \longrightarrow \mathcal{T}_{b,u}, \qquad \Phi(S, \mathbf{u}) = \text{insert}(S, \mathbf{u}).$$

Every maximal unary chain lies above a unique surviving skeleton node, including the root. Contracting all such chains and recording their lengths in preorder therefore gives

$$\Phi^{-1}(T) = \big(\mathrm{contract}(T), \mathrm{chainlengths}(T)\big), \qquad \Phi^{-1} \circ \Phi = \mathrm{id}, \quad \Phi \circ \Phi^{-1} = \mathrm{id}.$$

Both operations preserve child order and the prescribed node counts.

Uniform unary allocation. For $K > 1$, the composition sampler selects $K - 1$ separator positions uniformly:

$$1 \le t_1 < \cdots < t_{K-1} \le u + K - 1.$$

The forward and inverse maps are

$$u_1 = t_1 - 1, \qquad u_j = t_j - t_{j-1} - 1 \quad (2 \le j < K),$$

$$u_K = u + K - 1 - t_{K-1}, \qquad t_j = j + \sum_{\ell=1}^{j} u_\ell \quad (1 \le j < K).$$

Hence

$$u_j \ge 0, \qquad \sum_{j=1}^{K} u_j = u,$$

$$|\mathcal{U}_{u,K}| = \binom{u + K - 1}{K - 1} = W(u, K), \qquad P(\mathbf{u} \mid u, K) = \frac{1}{W(u, K)}.$$

For $K = 1, \mathcal{U}_{u,1} = \{(u)\}$ and $W(u, 1) = 1$. By the bijection and independence,

$$|\mathcal{T}_{b,u}| = |\mathcal{S}_b|\,|\mathcal{U}_{u,K}| = C_b W(u, K),$$

$$P(T \mid b, u) = P(S \mid b) P(\mathbf{u} \mid u, K) = \frac{1}{C_b} \frac{1}{W(u, K)},$$

$$\sum_{T \in \mathcal{T}_{b,u}} P(T \mid b, u) = 1.$$

Conditioning on the root arity. For a unary root ($u \ge 1$), reserving one root node gives the bijection

$$\{\mathbf{u} \in \mathcal{U}_{u,K} : u_1 \ge 1\} \longleftrightarrow \mathcal{U}_{u-1,K},$$
$$(u_1, u_2, \ldots, u_K) \longmapsto (u_1 - 1, u_2, \ldots, u_K),$$
$$W_{\mathrm{unary}} = W(u - 1, K) = \binom{u + K - 2}{K - 1}.$$

For a binary root ($b \ge 1$), excluding the root position gives

$$\{\mathbf{u} \in \mathcal{U}_{u,K} : u_1 = 0\} \longleftrightarrow \mathcal{U}_{u,K-1},$$
$$(0, u_2, \ldots, u_K) \longmapsto (u_2, \ldots, u_K),$$
$$W_{\mathrm{binary}} = W(u, K - 1) = \binom{u + K - 2}{K - 2}.$$

The inverses add one to the first entry or prepend zero, respectively. Let $\mathcal{T}_{b,u,r}$ contain the shapes with root arity r. Each admissible vector has probability $1/W_r$, independently of the skeleton:

$$|\mathcal{T}_{b,u,r}| = C_b W_r, \qquad P(T \mid b, u, r) = \frac{1}{C_b W_r}, \qquad \sum_{T \in \mathcal{T}_{b,u,r}} P(T \mid b, u, r) = 1.$$

The feasible boundary cases are

(b, u)	admissible allocation	result
$b = 0,\ u > 0$	(u)	one unary chain
$b > 0,\ u = 0$	$(0, \ldots, 0)$	unchanged binary skeleton
$b = u = 0$	(0)	one leaf.

A unary root with $u = 0$ or a binary root with $b = 0$ is infeasible and excluded. □

These statements characterize the raw, unlabeled shape proposal. The empirical draw of (N, B) controls how often different arity classes occur; Catalan and composition weights make the draw uniform within the selected class and root constraint. Subsequent operator/leaf labels, simplification, and acceptance checks determine the distribution of the final formulas.

Table 9: Base operator probabilities after the unary floor and before forced assignments or contextual exclusions. `Neg`, `Inv`, `Pow2`, and `Pow3` denote negation, reciprocal, square, and cube.

Binary operator	Probability	Unary operator	Probability
`Add`	0.219895438	`Neg`	0.034145627
`Sub`	0.114883686	`Inv`	0.015434137
`Mul`	0.509631078	`Abs`	0.012666583
`Div`	0.135253958	`Sqrt`	0.117837361
`Pow`	0.020335839	`Pow2`	0.278049997
		`Pow3`	0.073586740
		`Exp`	0.110216614
		`Log`	0.032029858
		`Sin`	0.200413219
		`Cos`	0.113221225
		`Tan`, `Arcsin`, `Arccos`	0.001000000 each
		`Arctan`, `Sinh`, `Cosh`	0.001000000 each
		`Tanh`	0.003398639
		`Cot`, `Sec`, `Csc`	0.001000000 each

B.3 Variables, constants, and exponents

1. **Variable coverage and ordinary leaves.** Reserve exponent leaves, shuffle ordinary leaves, and place each of the v variables once. Each remaining leaf is a constant with probability 0.702974962, or a variable drawn uniformly from $\{x_1, \ldots, x_v\}$.
2. **Constant magnitudes and signs.** Select a bin from Table 10, returning zero or sampling $\log_{10}|c|$ uniformly. With probability 0.151283414, replace the nonzero draw by a uniformly chosen matching value from $\{0.5, 1, 2, e, \pi\}$, if available. Positive and negative signs have probabilities 0.667642615 and 0.332357385, respectively. Exclude zero for leaves directly under `Mul`, `Div`, `Pow`, `Inv`, or `Log`.
3. **General-power exponents.** For a `Pow` leaf exponent, draw uniformly from $\{4, 5, 6\}$ with probability 0.536287857, uniformly from $\{0.25, 0.75, 1.25, 1.5, 2.5, 3.5\}$ with probability 0.062566906, or from $\mathcal{U}(0.1, 4)$ with probability 0.401145236. Negative exponents and aliases $-1, 0.5, 2, 3$ are excluded; reciprocal, square-root, square, and cube forms use unary operators.

Table 10: Constant-magnitude proposals. The smallest nonzero bin has the finite sampling range $[10^{-100}, 10^{-12})$. The original mass above 10^6 is redistributed proportionally between the two smallest nonzero bins.

$\lvert c\rvert$ interval	Probability	$\lvert c\rvert$ interval	Probability
0	0.000248984	$[10^{-1}, 1)$	0.297092815
$(0, 10^{-12})$	0.037271548	$[1, 10)$	0.313650602
$[10^{-12}, 10^{-6})$	0.062787186	$[10, 10^3)$	0.059584268
$[10^{-6}, 10^{-3})$	0.046988624	$[10^3, 10^6)$	0.011093352
$[10^{-3}, 10^{-1})$	0.171282619	$[10^6, \infty)$	0

B.4 Simplification and acceptance checks

1. **Processing order.** Reparse serialized expressions, fold numerical subtrees bottom-up, and apply the whitelist until stable, rejecting after 16 unsuccessful passes. Process targets before pairing, random candidates after variable mapping, and edited candidates after editing, all before TED labeling.
2. **Numerical subtrees and domains.** Require real, finite scalar constants with $|c| < 10^6$. Reject zero denominators, `Inv`(0), $\log(c)$ for $c \leq 0$, $\sqrt{c}$ for $c < 0$, inverse sine/cosine for $|c| > 1$, 0^c for $c \leq 0$, and negative constant bases with noninteger powers. Folded fitted constants remain fitted; rewrite-introduced structural constants are not fitted.

3. **Whitelist of algebraic rewrites.** The implemented whitelist contains the following groups:
   - *Identities and signs:* remove additive zeros, subtractive zero, multiplicative zero or one, and division by one; map $0 - x$ to $-x$, multiplication or division by -1 to negation, $x - x$ to 0, $x + x$ to $2x$, and xx to $\mathrm{Pow2}(x)$. Remove double negation and double inversion, collapse repeated absolute value, and simplify negative numerical literals.
   - *Collection of terms and factors:* flatten addition/multiplication components and combine constants, signed coefficients of like terms, and repeated factors. Group modulo associativity and commutativity, retaining first-occurrence order. Add positive integer exponents of repeated bases, without distribution or general fraction cancellation.
   - *Powers and absolute values:* apply $x^0 \mapsto 1$, $x^1 \mapsto x$, $1^x \mapsto 1$, and aliases $(-1, 0.5, 2, 3) \mapsto (\mathrm{Inv}, \mathrm{Sqrt}, \mathrm{Pow2}, \mathrm{Pow3})$. Combine nested positive integer powers by multiplying exponents; map $\sqrt{x^2}$ to $|x|$, $|-x|$ to $|x|$, $|x^{2k}|$ to x^{2k} for positive integer k, and $(-x)^2$ or $|x|^2$ to x^2.

   The whitelist excludes general distribution, factorization, rational cancellation, trigonometric identities, exponential/logarithmic cancellation, and unrestricted real-power identities.
4. **Operator-nesting restrictions.** The trigonometric family includes `Sin`, `Cos`, `Tan`, `Cot`, `Sec`, `Csc`, `Arcsin`, `Arccos`, `Arctan`, `Sinh`, `Cosh`, and `Tanh`. After folding, reject nested family members or any ancestor–descendant combination with `Exp`; also reject direct nesting within $\{\mathrm{Exp}, \mathrm{Log}\}$ or $\{\mathrm{Pow}, \mathrm{Pow2}, \mathrm{Pow3}, \mathrm{Sqrt}\}$. General powers require finite, positive numerical leaf exponents.
5. **Formula and retry limits.** Require a variable, at most 100 nodes and 512 prefix tokens, candidate variables drawn from the target, and numerical validity. Retry limits are 200 structural attempts, 200 tokenization calls, 100 numerical-validity calls, and 100 pair attempts with fixed source/edit family. Archived pair retries may redraw target length; the current generator instead holds raw length fixed for up to 200 filtered attempts. Simplification can shorten either output.

B.5 Target–candidate construction and local edits

1. **Independent and edited pairs.** Random pairs sample target and candidate independently; edited pairs modify a target copy. Map candidate and donor variables consistently into the target set, uniformly without replacement when possible and with replacement otherwise. Finite generation requests a 50:50 source split up to rounding; continuous pools are generated separately. Training uses the mixture in Appendix B.7.
2. **Edit family, count, and total budget.** Draw family/count proposals from Table 11, truncate counts using Table 12, and renormalize. For target length L, total affected-node charges are capped at $\max(2, \lceil rL \rceil)$ using the listed r. Charges are one for point/unary edits, branch size plus connecting operator for branch edits, the larger subtree size for replacement, and removed nodes for promotion. These charges differ from TED.
3. **Location selection.** Choose uniformly among eligible locations. With probability 0.85, restrict to paths disjoint from earlier edits when available; otherwise use all eligible locations.

Table 11: Edit-family probabilities and count proposals conditional on an edited pair. Short-target caps are applied after these proposals.

Family	Probability	Count values: probabilities
Point relabeling	0.20	$1, 2, 3, 4$: $(0.15, 0.40, 0.30, 0.15)$
Unary insertion/deletion	0.15	$1, 2, 3, 4$: $(0.20, 0.40, 0.25, 0.15)$
Branch insertion/deletion	0.05	$1, 2, 3$: $(0.25, 0.45, 0.30)$
Subtree replacement	0.15	$1, 2, 3$: $(0.25, 0.45, 0.30)$
Crossover/promotion	0.05	$1, 2$: $(0.60, 0.40)$
Mixed families	0.40	$2, 3, 4$: $(0.10, 0.30, 0.60)$

Operations within each family. These probabilities are conditional on the family. Operators follow Table 9, excluding the original operator for relabeling.

Table 12: Edit-count caps by target length L and cumulative affected-node budget ratio r. Each piecewise list gives the largest permitted count.

Family	Maximum count	r
Point / unary	1 for $L \leq 5$; 2 for $6 \leq L \leq 20$; 3 for $21 \leq L \leq 50$; 4 otherwise	0.30
Branch / subtree	1 for $L \leq 7$; 2 for $8 \leq L \leq 30$; 3 otherwise	0.35
Crossover/promotion	1 for $L \leq 20$; 2 otherwise	0.20
Mixed	2 for $L \leq 5$; 3 for $6 \leq L \leq 15$; 4 otherwise	0.40

1. **Point relabeling.** Choose numerical-value, variable, unary-operator, binary-operator, or variable-to-constant replacement with probabilities $(0, 0.10, 0.40, 0.45, 0.05)$. Variables change uniformly to another target variable. Operator changes preserve children and arity and require a variable-containing subtree. Variable-to-constant edits use the ordinary constant sampler and must retain a variable overall.
2. **Unary insertion or deletion.** With equal probability 0.5, wrap a subtree in a sampled unary operator or remove a unary operator while retaining its variable-containing child.
3. **Branch insertion or deletion.** Choose insertion/deletion with probability 0.5 each. For current size ℓ, define $c_{10}(\ell) = \max(1, \min(10, \lceil 0.10\ell \rceil))$. Insertion joins a subtree and a donor of at most $c_{10}(\ell)$ nodes under a sampled binary operator, with equally likely child order. Restrict donors to non-root subtrees with probability 0.70. Deletion retains a variable-containing child and removes a sibling of at most $c_{10}(\ell)$ nodes. Enforce the 100-node limit.
4. **Subtree replacement.** Select a subtree of size $s \leq c_{15}(\ell)$, with $c_{15}(\ell) = \max(1, \min(8, \lceil 0.15\ell \rceil))$, excluding the root unless the tree has one node. Matched, shorter, and longer donors have probabilities $(0.70, 0.05, 0.25)$ and size intervals
$$[\max(1, \lfloor 0.8s \rfloor),\ \max(1, \lfloor 0.8s \rfloor, \lceil 1.2s \rceil)],$$
$$[1,\ \max(1, s-1)],$$
$$[s+1,\ \min(c_{15}(\ell)+2, s+4)].$$
Cap upper endpoints at $100 - (\ell - s)$. Choose uniformly among eligible subtrees of a sampled donor formula, restricting to non-root donors with probability 0.70. Both subtrees must contain a variable; reject identical replacements.
5. **Crossover or promotion.** Crossover (probability 0.80) uses the preceding cap and intervals, size-mode probabilities $(0.80, 0.10, 0.10)$, and non-root donors. Promotion (probability 0.20) uniformly chooses an eligible ancestor–descendant pair, retains the variable-containing descendant, and removes 1 to $\max(1, \lceil 0.20\ell \rceil)$ nodes. Apply the same disjoint-path preference.
6. **Mixed operations.** For $m = 2$, select two distinct families; for $m = 3$, select two/three with probabilities $(0.75, 0.25)$; for $m = 4$, select two/three/four with probabilities $(0.60, 0.30, 0.10)$. Sample without replacement using point, unary, branch, subtree, and crossover/promotion weights $(0.30, 0.25, 0.20, 0.20, 0.05)$. Fill remaining slots with those weights, allowing repeats except for subtree and crossover/promotion, then shuffle.
7. **Retries, final validation, and labels.** Allow 30 sequence attempts and 10 attempts per operation. Proposal routines allow 12 point, 8 unary, 12 branch, and 20 subtree trials; donor search allows 40 formulas. Retry failed pairs with the same edit family. Reject candidates that normalize back to the target, lose variables, violate constraints, or fail numerical validation. Then compute normalized multi-view TED (Appendix A), rejecting nonfinite labels without targeting a distance interval. Command defaults allow a 40-second TED deadline, 200-second hard watchdog, and five failed task attempts before skipping.

B.6 Numerical observation sampling

The archived run uses the base mixture sampler adapted from MDLformer. Let v denote the number of target variables and M the requested point count.

1. **Mixture parameters.** Draw C uniformly from $\{1, \ldots, 9\}$, weights $a_k \sim \mathcal{U}(0,1)$, and $\rho_k = a_k / \sum_j a_j$. Each component draws $\boldsymbol{\mu}_k \sim \mathcal{N}(0, I_v)$, $c_{kj} \sim \mathcal{U}(0,1)$, and a random special-orthogonal rotation R_k for $v > 1$ (identity for $v = 1$). Allocate points by $\mathrm{Multinomial}(M; \boldsymbol{\rho})$.
2. **Component type and coordinates.** Independently for each component, choose Gaussian or uniform sampling with probability 0.5 each:

$$\mathbf{x} = \mathbf{z}R_k, \qquad \begin{cases} \mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_k, \mathrm{diag}(\mathbf{c}_k)), & \text{Gaussian,} \\ z_j = \mu_{kj} + \sqrt{c_{kj}}\, u_j, \quad u_j \sim \mathcal{U}(-1,1), & \text{uniform.} \end{cases}$$

   Gaussian and uniform coordinate variances are c_{kj} and $c_{kj}/3$, respectively. Rotate means and offsets together. The archived sampler leaves X unstandardized and uses no affine augmentation.
3. **Evaluation, filtering, and repeated draws.** Evaluate the target and discard nonfinite outputs or $|y| > 9.999 \times 10^{100}$. Draw up to five fresh mixture batches until M valid rows are available, then shuffle and retain M. Table construction allows 20 such calls and requires finite inputs/outputs. Failed stratified requests retry up to 32 rows from the same source/bin. Pair-generation checks use $M = 256$.
4. **Inputs supplied to the verifier.** Use 256 table rows, pad unused variable columns to 10, and normalize outputs by

$$\widetilde{y}_i = (y_i - \overline{y})/(\mathrm{std}(y) + 10^{-6}).$$

   Compute correlation summaries from 2,048 valid observations and use their first 256 shuffled rows for the table encoder. Each transformed correlation requires three valid points; mask invalid summaries (Appendix C). Observations come from the target; symbolic inputs come from the candidate.

B.7 TED-BALANCED TRAINING DRAWS AND VALIDATION

1. **Distance coverage and source probabilities.** Draw each of 12 TED bins with probability $1/12$. Given bin i, choose edited or random pairs with probabilities q_i and $1 - q_i$, respectively (Table 13):

$$P(i, \mathrm{edit}) = \frac{q_i}{12}, \qquad P(i, \mathrm{random}) = \frac{1 - q_i}{12}.$$

   Marginal source probabilities are 0.45 edited and 0.55 random. Equal bin mass leaves within-bin distances distributed as in the pool; the last two bins are wider. The main text's 0.6 boundary indicates the predominant source.
2. **How the mixture weights were fixed.** The August 28 calibration scan contained 18,343,472 edited and 52,541,662 random rows, with valid TED labels throughout. For the four mixed-source bins, use counts E_i, R_i to set

$$q_i = \frac{\lambda E_i}{\lambda E_i + R_i}, \qquad \lambda = 3.0244780430884326.$$

   Other bins use the table's zero/one choices. These calibration counts precede the final pool snapshot.
3. **Row selection, batches, and reproducibility.** Draw bin/source requests independently. Traverse each source/bin by deterministic permutations without replacement, restarting when exhausted. Seed 42 and update number fix the global plan; replay category draws on resume. Use four workers, 32 examples per microbatch, and two accumulation rounds (global batch 256). Expected counts are $256/12$ per bin, 115.2 edited, and 140.8 random. Contrastive pretraining instead samples sources equally.
4. **Reserved rows and fixed validation set.** Reserve the first 32 rows per source/bin from stratified training. Fixed validation contains 32 pairs per bin: 384 total, comprising 173 edited and 211 random pairs.

Bayesian interpretation of the source choice. Let $Z \in \{\text{edit}, \text{random}\}$ be the pair source and $I \in \{1, \dots, 12\}$ its TED bin. Let $E = \sum_i E_i > 0$ and $R = \sum_i R_i > 0$ be the calibration source totals. The empirical bin likelihoods are $P(I = i \mid Z = \text{edit}) = E_i/E$ and $P(I = i \mid Z = \text{random}) = R_i/R$.

Proposition 6 (Bayesian source sampling and calibration across TED bins)**.** *For an edited-source prior $P(Z = \text{edit}) = \rho \in (0, 1)$ and any bin of positive mixture probability, Bayes' rule gives*

$$P(\text{edit} \mid i) = \frac{\rho E_i/E}{\rho E_i/E + (1-\rho)R_i/R} = \frac{\lambda E_i}{\lambda E_i + R_i}, \qquad \lambda = \frac{\rho R}{(1-\rho)E}. \tag{22}$$

Conversely, any $\lambda > 0$ corresponds to $\rho = \lambda E/(\lambda E + R)$.

Set $q_i = 1$ for $i = 1, 2, 3$, use this posterior for $i = 4, 5, 6, 7$, and set $q_i = 0$ for $i = 8, \dots, 12$. Drawing I uniformly and then the source with edited probability q_I produces

$$\begin{aligned} Q(I = i) &= \frac{1}{12}, \\ Q(Z = \text{edit}) = F(\lambda) &= \frac{1}{12}\left(3 + \sum_{i=4}^{7} \frac{\lambda E_i}{\lambda E_i + R_i}\right). \end{aligned} \tag{23}$$

If $E_i, R_i > 0$ in the four mixed bins, every edited marginal in $(3/12, 7/12)$, including 0.45, determines a unique $\lambda > 0$.

Proof. **Posterior source law.** For a bin of positive mixture mass, the joint and marginal probabilities are

$$\begin{aligned} P(Z = \text{edit}, I = i) &= \rho E_i/E, \\ P(Z = \text{random}, I = i) &= (1-\rho)R_i/R, \\ P(I = i) &= \rho E_i/E + (1-\rho)R_i/R > 0. \end{aligned}$$

Therefore

$$\begin{aligned} P(\text{edit} \mid i) &= \frac{P(Z = \text{edit}, I = i)}{P(I = i)} = \frac{\rho E_i/E}{\rho E_i/E + (1-\rho)R_i/R} \\ &= \frac{[\rho R/((1-\rho)E)]E_i}{[\rho R/((1-\rho)E)]E_i + R_i} = \frac{\lambda E_i}{\lambda E_i + R_i}, \\ P(\text{random} \mid i) &= \frac{R_i}{\lambda E_i + R_i}, \qquad P(\text{edit} \mid i) + P(\text{random} \mid i) = 1. \end{aligned}$$

The prior–multiplier correspondence is

$$\lambda = \frac{\rho R}{(1-\rho)E} \iff \lambda E = \rho(\lambda E + R) \iff \rho = \frac{\lambda E}{\lambda E + R} \in (0, 1), \qquad \lambda > 0.$$

In particular, the source-prior odds are $\rho/(1-\rho) = \lambda E/R$.

Training marginals. Under uniform bin sampling and the prescribed q_i,

$$\begin{aligned} Q(I = i, Z = \text{edit}) &= \frac{q_i}{12}, & Q(I = i, Z = \text{random}) &= \frac{1 - q_i}{12}, \\ Q(I = i) &= \frac{q_i + (1 - q_i)}{12} = \frac{1}{12}, & \sum_{i=1}^{12} Q(I = i) &= 1. \end{aligned}$$

Thus

$$\begin{aligned} Q(Z = \text{edit}) &= \sum_{i=1}^{12} \frac{q_i}{12} \\ &= \frac{1}{12}\left(\underbrace{\sum_{i=1}^{3} 1}_{3} + \sum_{i=4}^{7} \frac{\lambda E_i}{\lambda E_i + R_i} + \underbrace{\sum_{i=8}^{12} 0}_{0}\right) = F(\lambda). \end{aligned}$$

The zero/one choices are imposed source restrictions; the Bayesian posterior is used only in the four mixed bins.

Existence and uniqueness of calibration. With $E_i, R_i > 0$ for $i = 4, \dots, 7$,

$$\frac{d}{d\lambda}\frac{\lambda E_i}{\lambda E_i + R_i} = \frac{E_i(\lambda E_i + R_i) - \lambda E_i^2}{(\lambda E_i + R_i)^2} = \frac{E_i R_i}{(\lambda E_i + R_i)^2} > 0,$$

$$F'(\lambda) = \frac{1}{12}\sum_{i=4}^{7} \frac{E_i R_i}{(\lambda E_i + R_i)^2} > 0.$$

Each denominator is positive on $(0, \infty)$, so F is continuous there, and

$$\lim_{\lambda \downarrow 0} \frac{\lambda E_i}{\lambda E_i + R_i} = 0, \qquad \lim_{\lambda \to \infty} \frac{\lambda E_i}{\lambda E_i + R_i} = 1,$$

$$\lim_{\lambda \downarrow 0} F(\lambda) = \frac{3}{12}, \qquad \lim_{\lambda \to \infty} F(\lambda) = \frac{7}{12}.$$

For any target $\tau \in (3/12, 7/12)$, these limits provide

$$0 < \lambda_- < \lambda_+ < \infty, \qquad F(\lambda_-) < \tau < F(\lambda_+).$$

Continuity gives a solution in (λ_-, λ_+), while

$$\lambda_1 < \lambda_2 \implies F(\lambda_1) < F(\lambda_2)$$

gives uniqueness. Therefore

$$\forall \tau \in (3/12, 7/12), \quad \exists! \lambda > 0 : \ F(\lambda) = \tau.$$

Since $3/12 < 0.45 < 7/12$, the reported target has a unique multiplier. □

For the reported calibration counts, $\lambda \simeq 3.0244780430884326$ gives the target edited marginal 0.45. This final marginal follows uniform bin sampling and need not equal the auxiliary source prior ρ.

Table 13: TED-bin probabilities, calibrated source choices, and the counts used to determine them. Every bin has training probability $1/12$; counts describe the August 28 calibration scan.

TED interval	Edited count E_i	Random count R_i	q_i (edit)	$1 - q_i$ (random)
$[0, 0.1)$	1,562,156	106,540	1	0
$[0.1, 0.2)$	2,578,355	148,986	1	0
$[0.2, 0.3)$	3,802,093	478,678	1	0
$[0.3, 0.4)$	3,536,928	1,073,557	0.908795814	0.091204186
$[0.4, 0.5)$	2,699,071	2,321,306	0.778598241	0.221401759
$[0.5, 0.6)$	2,159,613	6,491,726	0.501534776	0.498465224
$[0.6, 0.7)$	1,147,026	12,966,794	0.211071169	0.788928831
$[0.7, 0.8)$	405,094	9,157,137	0	1
$[0.8, 0.9)$	216,774	4,606,913	0	1
$[0.9, 1)$	37,601	1,838,357	0	1
$[1, 1.5)$	163,252	7,710,484	0	1
$[1.5, \infty)$	35,509	5,641,184	0	1

C VERIFIER ARCHITECTURE AND TRAINING

C.1 INPUTS AND REGRESSION FEATURES

The interface supports up to ten variables, at most 512 formula tokens, and at least three finite observations. It samples up to 256 rows for the table encoder and 2,048 rows for regression summaries, using a fixed inference-preprocessing seed. The table branch standardizes y by $(y - \mathrm{mean}(y))/(\mathrm{std}(y) + 10^{-6})$. Regression summaries use raw values and their specified transforms.

For each x_j, the 12 target/input transform pairs are

$$(\log y, \log x_j), \quad (y, x_j),\ (y, x_j^2),\ (y, x_j^3),\ (y, \sqrt{x_j}),\ (y, \log x_j),$$
$$(y, x_j^{-1}),\ (y, x_j^{-2}),\ (y, e^{x_j}),\ (y, \sin x_j),\ (y, \cos x_j),\ (y, \tan x_j). \tag{24}$$

Write the kth pair as $(g_k(y), h_k(x_j))$. For each variable j, let $\mathcal{I}_{jk}$ contain the rows on which both transformed values are finite and within their domains; the log–log pair, for example, requires $x_{ij} > 0$ and $y_i > 0$. With $u_i = h_k(x_{ij})$ and $v_i = g_k(y_i)$, we compute the signed Pearson correlation

$$r_{jk} = \frac{\sum_{i \in \mathcal{I}_{jk}} (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i \in \mathcal{I}_{jk}} (u_i - \bar{u})^2} \sqrt{\sum_{i \in \mathcal{I}_{jk}} (v_i - \bar{v})^2}},$$

where the means are over $\mathcal{I}_{jk}$. This equals $\operatorname{sign}(\widehat{\beta}_{jk})\sqrt{R^2_{jk}}$ for a one-predictor least-squares regression with an intercept, where $\widehat{\beta}_{jk}$ is its slope and R^2_{jk} its coefficient of determination. For numerical stability, each transformed vector is divided by its largest absolute value before centering, which leaves its correlation unchanged, and the result is clipped to $[-1, 1]$. A relation is masked if fewer than three valid rows remain, either transformed vector has zero variance, or the correlation is undefined; unused variables are also masked.

Each valid (j, k) relation produces a token. Let $e_{\text{out}}(g_k)$, $e_{\text{pair}}(k)$, and $e_{\text{var}}(j)$ be learned 64-dimensional embeddings for the output form (y or $\log y$), the relation identity, and the input variable. The 12 relation identities are distinct, including the two relations that use $\log x_j$. The descriptor and gated token are

$$q_{jk} = [e_{\text{out}}(g_k); e_{\text{pair}}(k); e_{\text{var}}(j); r_{jk}] \in \mathbb{R}^{193},$$
$$t_{jk} = |r_{jk}|(W_R q_{jk} + b_R) \in \mathbb{R}^{768}.$$

The projection W_R, b_R is shared across relations. The signed r_{jk} enters the descriptor, while $|r_{jk}|$ reduces the magnitude of weak-correlation tokens. Masked tokens are excluded from the learned attention pooling. For the valid set $\mathcal{V}$, the pooled regression representation is

$$\alpha_{jk} = \frac{\exp(w^\top t_{jk} + b)}{\sum_{(p,q) \in \mathcal{V}} \exp(w^\top t_{pq} + b)}, \qquad z_R = \sum_{(j,k) \in \mathcal{V}} \alpha_{jk} t_{jk}.$$

Here w, b are learned attention-score parameters. If no relation is valid, the pooled vector is zero.

The data encoder embeds sign/mantissa/exponent tokens into 64 dimensions, concatenates each padded row, and projects it to width 768. The formula encoder adds token, tree-address, and depth embeddings. Both use six Transformer layers, six attention heads, learned attention pooling, and output projection. SwiGLU hidden width is 2,048. Part of its value bias is tied to match the parameter count of the original four-times-width feed-forward block.

C.2 Optimization

For matched Stage 1 data/formula embeddings (u_i, v_i), the loss is

$$\mathcal{L}_{\text{align}} = -\frac{1}{2B} \sum_i \left[\log \frac{\exp(u_i^\top v_i / t)}{\sum_j \exp(u_i^\top v_j / t)} + \log \frac{\exp(v_i^\top u_i / t)}{\sum_j \exp(v_i^\top u_j / t)} \right], \tag{25}$$

with temperature $t = 1$ and raw, unnormalized dot products. The paired expression is the generating target. Stage 2 initializes both encoders, adds the regression branch and fusion head, and updates all parameters using Equation 9. The regression head does not clip outputs.

Stage 2 uses AdamW with $\beta = (0.9, 0.95)$, weight decay 0.01, gradient clipping at 1.0, and microbatches of 32. Validation is evaluated every 100 updates. The selected Stage 2 checkpoint has archived validation objective 0.12612656.

The bins are $[0, 0.1), \ldots, [0.9, 1.0), [1.0, 1.5), [1.5, \infty)$, sampled equally. Conditional probabilities of drawing an edited pair are

$$(1,\ 1,\ 1,\ 0.9088,\ 0.7786,\ 0.5015,\ 0.2111,\ 0,\ 0,\ 0,\ 0,\ 0).$$

Their mean is 0.45, with complementary random-pair probabilities. Validation draws a fixed source mixture with 32 examples per bin, totaling 384 examples. The reserved source/bin rows and final mixed validation batch are distinct.

Table 14: Completed verifier training configuration. Checkpoints were selected on synthetic validation, not benchmark test scores.

Setting	Stage 1	Stage 2
Global batch size	256	256
Updates	55,000	100,000
Learning rate	10^{-4} peak	2×10^{-5} to 2×10^{-6}
Schedule	Warmup–stable–decay	Warmup + cosine decay
Warmup updates	2,000	2,000
Stable / decay updates	48,000 / 5,000	—
Seed / training GPUs	42 / 4	42 / 4
Selected checkpoint update	54,500	86,600

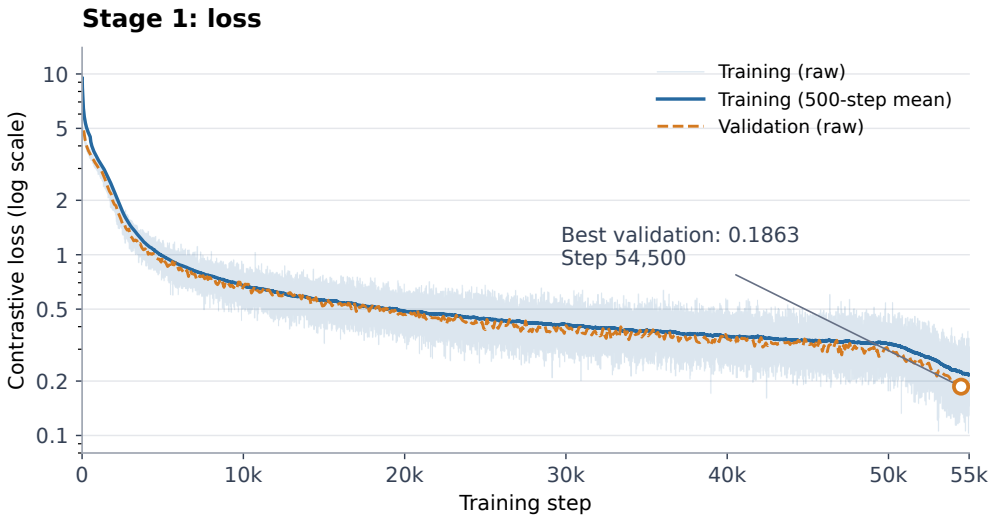


Figure 10: Stage 1 training and validation loss.

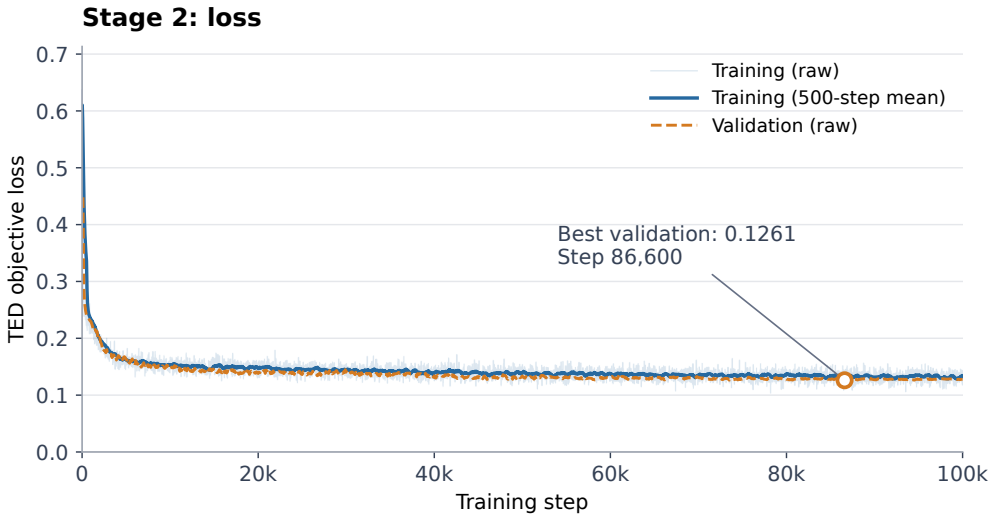


Figure 11: Stage 2 training and validation loss.

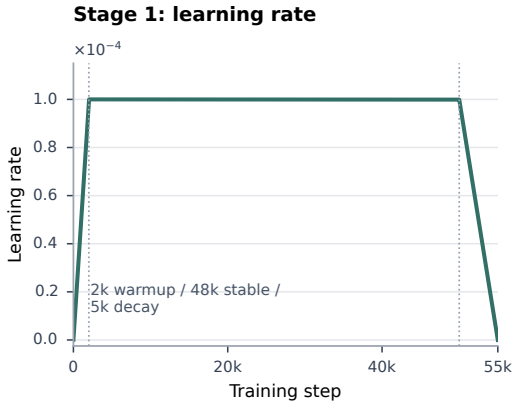


Figure 12: Stage 1 learning-rate schedule.

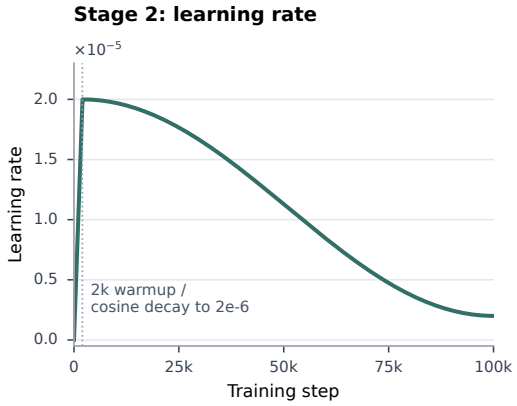


Figure 13: Stage 2 learning-rate schedule.

D SEARCH IMPLEMENTATION AND ABLATION CONTROLS

Algorithm 1 L2G-SR: symbolic search with learned structural feedback

Require: Observations $\mathcal{D}$, metadata, frozen V_θ; proposal budget
1: Split $\mathcal{D}$; initialize four islands
2: **for** each round in the balanced M/T schedule **do**
3: Select island and parent by validation MSE or predicted TED
4: Analyst uses diagnostics and history; generator proposes four formulas
5: For each proposal: validate, fit, simplify, refit, and score by MSE and V_θ
6: Add valid formulas; deduplicate and prune by both scores
7: **end for**
8: **return** simplest valid formula within the validation-MSE tolerance

Table 15: Search settings shared by TED and matched numerical-only conditions.

Setting	Value
Backbone for both roles / temperature	Llama3.1-8B-Instruct / 0.8
Islands / capacity / M and T quotas	4 / 64 / 32 each
Initialization / rounds / children	40 / 90 / 4
M rounds / T rounds	45 / 45
Tournament sizes, M / T	3 / 2, with replacement
Credibility threshold / exploration cap	Validation NMSE ≤ 1 / 8
Validation fraction / maximum fit rows	0.2 / 4,096
Independent constants / BFGS starts	At most 10 / 3
BFGS gradient tolerance	10^{-5}
Per-slot numerical budget	30 seconds; 6,000 objective evaluations
Postprocessing passes	At most 16
Analyzer tool calls / interactions	At most 8 / 4 per round
Analyzer / generator input-token caps	6,000 / 4,000
Analyzer / generator output-token caps	2,200 / 1,800

When subtraining exceeds 4,096 rows, fitting observations are sampled by target-value strata. Constants are unbounded real values; the fitting budget bounds computation, not coefficients. Mapped parent parameters provide warm starts. Linear-in-parameter formulas can use a scaled least-squares initialization. Supported symbolic derivatives supply gradients, with numerical differentiation as a fallback.

A ratio above 10^4 between the 95th and 5th percentiles of $|y|$, or its maximum and median, activates transformed fitting. Positive targets use log residuals; otherwise the implementation uses $\operatorname{asinh}(y/s_y)$. To screen term removal before refitting, we set each fitted constant θ_j to zero in the

current formula f, giving $f_{j\to 0}$. We consider the contribution small only when, over all n search observations,

$$\Delta_i^{(j)} = \frac{f_{j\to 0}(x_i) - f(x_i)}{\max\{|y_i|, 10^{-6} s_y\}},$$
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\Delta_i^{(j)}\right)^2} \le 10^{-4}, \qquad \max_i \left|\Delta_i^{(j)}\right| \le 10^{-3}, \tag{26}$$

where $s_y = \max\{\mathrm{median}_i\,|y_i|,\, 10^{-6} Q_{0.75}(|y|),\, \epsilon_{\mathrm{tiny}}\}$ is the robust target scale. Other proposed edits use the same normalized output change before refitting. After refitting, the proposal must satisfy, separately on fixed subtraining and validation,

$$\mathrm{MSE}_{\mathrm{new}} \le \mathrm{MSE}_{\mathrm{old}} + 10^{-3}\mathrm{MSE}_{\mathrm{old}} + 10^{-12}\,\mathrm{Var}(y).$$

These checks are untrimmed. Symbolic-equivalence proposals use stricter output-change checks. The verifier scores only the final accepted effective expression.

Let m be the best validation MSE and σ_y^2 the target variance used by final selection. Eligible candidates satisfy

$$\mathrm{MSE}(f) \le m + \max\{10^{-10}\max(\sigma_y^2, 10^{-30}), 10^{-4} m\}.$$

Among them, selection minimizes complexity, parameter count, MSE, and then a stable identifier. Tournament ties are broken uniformly. Unavailable structural predictions trigger a random-valid-candidate fallback for the T tournament.

The no-TED control substitutes MSE in the compatibility score field used by comparisons and retention, and labels this substitution in the prompts. The schedule and tournament sizes remain unchanged. It performs no TED model loading or prediction; any offline true-distance computation cannot affect search decisions.

E Detailed completed results

E.1 Aggregation and reporting

We aggregate completed equations within each domain/seed and then report

$$\bar{m} = \frac{1}{3}\sum_{s=1}^{3} m_s, \qquad \mathrm{SD}(m) = \sqrt{\frac{1}{2}\sum_{s=1}^{3}(m_s - \bar{m})^2}.$$

Accuracy dispersion is in percentage points. Entries taken from DE preserve its reported precision and aggregation. Our RESTART reproduction reports sample SD across its own three seeds.

The task configuration totals 240 equations. We report the actual evaluated domain counts. For seed 347, Chemistry contributes 35 completed equations to its domain mean; both other seeds cover all 36 Chemistry equations.

The float64 evaluator discards $\lfloor 0.05n \rfloor$ points per equation. Nonfinite predictions consume this allowance before finite points are ordered by relative error. Exceeding the allowance makes the equation invalid. The retained-set target mean is recomputed for the NMSE denominator, protected by the smallest positive float64 value. This differs from untrimmed online validation.

Table 16: Sample standard deviations for the RESTART and SR-Scientist means in Table 1. Accuracy standard deviations are in percentage points.

	Transform		Physics		Materials		Chemistry		Biology	
Method	NMSE SD	$\mathrm{Acc}_{0.01}$ SD	NMSE SD	$\mathrm{Acc}_{0.01}$ SD	NMSE SD	$\mathrm{Acc}_{0.01}$ SD	NMSE SD	$\mathrm{Acc}_{0.01}$ SD	NMSE SD	$\mathrm{Acc}_{0.01}$ SD
Llama3.1-8B-Instruct										
RESTART	7.06e-3	1.88	3.47e-3	4.73	6.91e-5	6.11	1.23e-6	3.21	1.91e-2	6.36
SR-Scientist	1.94e-3	1.38	3.18e-3	1.31	1.49e-3	0.00	7.47e-5	2.78	4.53e-2	4.17
Qwen3-4B-Instruct-2507										
RESTART	1.08e2	1.04	5.48e-2	2.62	1.14e-3	4.00	3.81e-2	8.49	2.61e0	2.41

Table 17 compares our reproduced RESTART and SR-Scientist runs with L2G-SR on successful LLM usage. We count distinct successful API responses, including responses whose generated expressions are later rejected, and exclude failed requests. Input and output tokens are the provider-reported prompt and completion tokens for those responses. Each mean uses the same 480 completed problem runs per method; duplicate response IDs are removed from the RESTART and L2G-SR journals, while SR-Scientist uses its successful-response counters.

Table 17: Successful LLM calls and tokens per problem for the Llama3.1-8B-Instruct runs shared by the three methods.

Method	Successful calls per problem	Input tokens per problem	Output tokens per problem
RESTART (reproduced)	422.7	715,609	103,528
SR-Scientist (reproduced)	519.2	1,517,007	63,266
L2G-SR	189.2	847,822	48,130

E.2 In-distribution Oscillation results

Table 18 reports the in-distribution results on the two real-world Oscillation problems.

	Oscillation 1		Oscillation 2	
Method	NMSE ↓	$\mathrm{Acc}_{0.01}^{\mathrm{pt}}$ ↑	NMSE ↓	$\mathrm{Acc}_{0.01}^{\mathrm{pt}}$ ↑
		Llama3.1-8B-Instruct		
LLMDirect	6.38e-7	**100.00**	2.31e-5	88.79
	±4.34e-7	±0.00	±1.89e-5	±9.37
LLM-SR	1.21e-5	94.31	4.51e-4	63.97
	±1.67e-5	±9.86	±7.17e-4	±34.95
LaSR	9.61e-4	16.26	8.36e-4	31.37
	±3.66e-4	±3.35	±3.27e-4	±9.05
SGA	2.23e-4	68.78	4.74e-3	7.96
	±3.76e-4	±44.14	±2.15e-3	±1.71
RESTART	4.61e-6	99.70	4.11e-3	11.14
	±3.27e-6	±0.52	±1.66e-3	±1.28
SR-Scientist	8.80e-10	**100.00**	4.21e-4	74.55
	±7.63e-10	±0.00	±7.30e-4	±44.08
L2G-SR	**6.22e-30**	**100.00**	**3.61e-7**	**100.00**
	±7.03e-30	±0.00	±5.21e-7	±0.00

Table 18: ID results on Oscillation 1 and 2.

E.3 Out-of-distribution evaluation

We evaluate how the selected expressions generalize to shifted inputs.

Under this protocol, L2G-SR has the best displayed NMSE and accuracy in every domain for both backbones.

E.4 Oscillation out-of-distribution results

Table 20 reports the OOD results corresponding to the ID comparison in Table 18. We evaluate the same selected formulas without refitting on the 10,000 observations in each official OOD split. Metrics are scored in float64 after removing the 500 observations with the largest relative errors. Pointwise $\mathrm{Acc}_{0.01}^{\mathrm{pt}}$ is the percentage of the 9,500 retained observations whose relative error is at most 0.01. All entries are means and sample standard deviations across three seeds.

L2G-SR has the lowest mean OOD NMSE and highest mean pointwise accuracy on both problems. On Oscillation 2, its pointwise OOD accuracy is $91.24 \pm 9.41\%$. The large RESTART mean NMSE on this problem is dominated by seed 181, whose retained-set NMSE is 6.14×10^{20}. Seed 347 has one nonfinite prediction; this observation consumes one of the 500 allowed removals and the run remains included in the summary.

Table 19: OOD evaluation on LSR-Synth. Published baseline rows are retained as reported. Our reproduced rows show mean ± standard deviation across three seeds.

	Physics		Materials		Chemistry		Biology	
Method	NMSE ↓	$\mathrm{Acc}_{0.01}$ ↑	NMSE ↓	$\mathrm{Acc}_{0.01}$ ↑	NMSE ↓	$\mathrm{Acc}_{0.01}$ ↑	NMSE ↓	$\mathrm{Acc}_{0.01}$ ↑
Llama3.1-8B-Instruct								
A-SR	6.55e3	25.00	6.90e-4	**100.00**	2.06e0	25.00	1.08e0	45.83
RESTART	2.13e2	10.08	6.65e-3	90.28	7.79e-1	41.90	2.10e1	18.84
	±1.89e2	±4.84	±5.61e-3	±4.81	±2.44e-1	±4.36	±2.26e1	±6.64
SR-Scientist	7.70e1	21.71	9.09e-2	75.00	2.33e0	35.24	7.49e3	23.19
	±7.36e1	±2.69	±7.82e-2	±11.02	±2.53e0	±4.36	±1.30e4	±2.51
L2G-SR	**2.13e-3**	**90.70**	**2.27e-5**	**100.00**	**1.92e-1**	**69.22**	**1.99e-1**	**66.67**
	±1.42e-3	±2.33	±2.21e-5	±0.00	±8.96e-2	±7.27	±1.56e-1	±5.02
Qwen3-4B-Instruct-2507								
LLMDirect	3.43e4	6.82	1.94e-1	72.00	5.98e6	0.00	1.18e2	4.17
LLM-SR	8.17e4	9.09	2.75e-1	72.00	6.57e2	11.11	2.24e1	12.50
LaSR	1.08e4	9.09	3.25e-1	40.00	4.49e2	2.78	9.66e2	4.17
SGA	9.92e5	6.82	2.91e-1	40.00	1.55e3	2.78	1.83e5	4.17
DE	1.97e3	15.91	4.85e-2	80.00	1.09e1	11.11	1.14e1	12.50
A-SR	3.17e2	18.18	3.64e-3	96.00	8.49e-1	19.44	7.41e0	16.67
L2G-SR	**1.06e-2**	**89.15**	**1.51e-5**	**100.00**	**1.41e-1**	**63.81**	**1.08e-1**	**59.42**
	±1.37e-2	±1.34	±1.89e-5	±0.00	±4.11e-2	±1.65	±3.99e-2	±2.51

Table 20: OOD results on Oscillation 1 and 2. NMSE (↓) and pointwise $\mathrm{Acc}^{\mathrm{pt}}_{0.01}$ (↑, %) are computed after 5% relative-error trimming. Entries report mean ± sample standard deviation over three seeds. Bold and underlined means indicate the best and second-best displayed values, respectively.

	Oscillation 1		Oscillation 2	
Method	NMSE ↓	$\mathrm{Acc}^{\mathrm{pt}}_{0.01}$ ↑	NMSE ↓	$\mathrm{Acc}^{\mathrm{pt}}_{0.01}$ ↑
Llama3.1-8B-Instruct				
LLMDirect	1.03e-2	45.06	3.83e-3	52.51
	±0.88e-2	±18.89	±3.01e-3	±14.82
LLM-SR	3.95e-2	23.52	6.28e-3	29.80
	±4.49e-2	±33.64	±0.76e-3	±18.29
LaSR	1.98e-1	4.60	1.75e-2	13.99
	±2.44e-1	±4.91	±1.56e-2	±5.65
SGA	3.19e-1	37.22	2.30e-2	7.12
	±2.78e-1	±50.88	±1.46e-2	±3.86
RESTART	8.51e-3	15.10	2.05e20	1.15
	±6.89e-3	±19.34	±3.55e20	±1.33
SR-Scientist	3.88e-4	79.11	2.13e-2	65.70
	±4.14e-4	±15.35	±3.68e-2	±55.48
L2G-SR	**3.11e-29**	**100.00**	**5.95e-5**	**91.24**
	±3.48e-29	±0.00	±5.55e-5	±9.41

E.5 Baseline configurations and RESTART reproduction

For LLMDirect, LLM-SR, LaSR, SGA, and DE, Table 1 uses the Llama3.1-8B-Instruct results reported by DE (Pang et al., 2026). We preserve DE's reported configurations rather than rerunning these methods.

Reported baseline parameters. The imported runs share the following settings:

- **Backbone:** Llama3.1-8B-Instruct.
- **Generation temperature:** 0.8.

- **Maximum output length:** 8,192 tokens.
- **Inference system:** vLLM on 4 NVIDIA A800-SXM4-80GB GPUs.

LLMDirect.

- **Hypotheses per prompt:** 5.
- **Total LLM-generated candidates per problem:** 5.
- **Execution timeout per hypothesis:** 30 seconds.
- **Constant optimizer:** SciPy BFGS.

LLM-SR.

- **Equation programs per prompt:** 4.
- **Parallel evaluators:** 4.
- **Islands:** 10.
- **In-context parents per prompt:** 4.
- **Maximum parameters per equation skeleton:** 10.
- **Constant optimizer:** SciPy BFGS.
- **Total LLM-generated candidates per problem:** 1,000.

LaSR.

- **Iterations:** 25.
- **Evolutionary cycles per iteration:** 550.
- **Populations:** 10.
- **Population size:** 33.
- **Operators:** $+, -, \times, \div, \wedge, \exp, \log, \sqrt{\cdot}, \sin, \cos, \tan$, and $\cosh$.
- **LLM-operator weights:** `llm_mutate`=0.005, `llm_crossover`=0.005, and `llm_gen_random`=0.005.
- **Total LLM-generated candidates per problem:** 1,000.
- **Random non-LLM mutations per problem:** $> 453{,}000$.
- **Other PySR parameters:** defaults of the evaluated implementation; DE reports no further values.

SGA.

- **Feedback objective:** MSE.
- **Continuous-parameter optimizer:** `torch.optim.Adam`.
- **Differentiable model:** PyTorch `torch.nn.Module`.
- **Total LLM-generated candidates per problem:** 1,000.
- **Adam hyperparameters:** not reported by DE.

DE.

- **Islands:** 4; **capacity per island:** 400; **reset interval:** 50 generations.
- **Generations:** 100; **offspring per generation:** 4; **maximum LLM-generated candidates per problem:** 400.
- **Selection:** fitness-proportional Boltzmann sampling; **initial temperature:** 0.5; **cooling rate:** 0.95; **schedule:** exponential decay; **stagnation-recovery temperature:** 2.0.
- **Parents per prompt:** 1 for non-crossover operators and 2 for crossover.
- **Maximum refinement rounds per sample:** 4.

- **Reflective-memory update interval:** 12 generations; **breakthrough threshold:** 0.1.
- **Reflection context:** 3 elite examples and 3 failed examples.
- **Stagnation window:** 8 generations; **distilled insights per update:** 3.

The RESTART row in Table 1 uses our reproduction, rather than the original paper's reported scores. It evaluates all 240 equations with three seeds (720 valid task results).

Table 21: Settings for our RESTART reproduction.

Setting	Value
Backbone / temperature	Llama3.1-8B-Instruct / 0.8
Training / internal-validation fractions	0.8 / 0.2
Islands / generations / slots per generation	10 / 100 / 4
LLM-generated candidate budget	400 slots per problem
Refinement and memory	E2E-assisted refinement / concept library
Maximum fit rows / constants	4,096 / 10
BFGS starts / initial constant	3 / 1.0
BFGS iterations per parameter / gradient tolerance	200 / 10^{-5}
Per-slot fitting timeout / objective evaluations	30 seconds / 6,000
Parameter domain	Unbounded real values
Final selection	Minimum internal-validation MSE
Additional computation	Numerical/E2E initialization and E2E calls

The reproduction adapts the released island, refinement, and concept-library code to the same numerical-fitting interface used by L2G-SR. The 10 islands, E2E-based refinement, and concept library retain the method's native roles. Numerical/E2E initialization uses 0 LLM proposal slots, and E2E calls add computation beyond the 400-slot budget.

Generated Python candidates are converted to the shared restricted algebra without executing generated code. The converter preserves the generated formula structure. Fitting uses the shared BFGS backend and budgets; our post-fit expression-pruning search is not applied to RESTART candidates. Final selection uses minimum internal-validation MSE. The official test arrays are scored only after search completion, without refitting.

Table 22: Per-problem search-budget comparison. Candidate or proposal counts follow the accounting used by DE and our implementations; the final column records method-specific computation outside those counts.

Method	Counted LLM search budget	Additional search or optimization
LLMDirect	5 generated candidates	BFGS constant refinement for each hypothesis.
LLM-SR	1,000 generated candidates	Island evolution and BFGS constant refinement; these operations act on the counted candidates.
LaSR	1,000 LLM-generated candidates	More than 453,000 random non-LLM mutations, in addition to PySR fitting and selection.
SGA	1,000 generated candidates	Adam-based inner-loop optimization of continuous parameters.
DE	At most 400 generated candidates (100×4)	Diagnostic tools, BFGS constant fitting, and reflective-memory updates are outside the candidate count.
RESTART (ours)	400 proposal slots (100×4)	Numerical/E2E initialization and E2E calls do not consume proposal slots; constant fitting follows the shared BFGS limits.
L2G-SR	400 charged proposal slots ($40 + 90 \times 4$)	

LLMDirect is a Best-of-5 baseline. DE, our RESTART reproduction, and L2G-SR use 40% of the 1,000-candidate LLM budget assigned to the iterative LLM-SR, LaSR, and SGA baselines. LaSR additionally evaluates more than 453 non-LLM mutations per counted LLM candidate. RESTART and L2G-SR match at 400 nominal proposal slots.

E.6 VERIFIER ABLATION DETAILS

Each variant in Table 5 uses training seed 42, Stage 2 batch size 256, and 100,000 updates. Datapool snapshots, the 12-bin sampler, the loss in Equation 9, validation composition, and optimization settings match the full verifier. The fixed validation set contains 173 edited and 211 random pairs, with 32 pairs per TED bin. Among its 384 pairs, 327 have $d_{\mathrm{str}} \leq 1$; the near-target MAE and RMSE use only those 327 pairs. Global MAE uses all pairs.

Without tree address, the formula encoder still receives prefix tokens, but its depth/address embedding contribution is zero in both training stages. The original position parameters remain allocated, giving the same total parameter count as the full model, although the zero-gated parameters have no effect. Formula only reuses the full model's Stage 1 formula encoder, retains its tree-address inputs, and trains only that encoder and the fusion head. Its numerical encoders are absent, so its parameter count is smaller. Without the correlation branch, the data and formula encoders are initialized from the same full-model Stage 1 checkpoint, but the signed-correlation encoder is absent and its fusion slot is fixed to zero. One-stage retains all full-model inputs and parameters, uses no Stage 1 checkpoint, and trains from random initialization for the same 100,000 updates under the Stage 2 objective.

We select all reported checkpoints by the same validation objective.

E.7 CANDIDATE-LEVEL VERIFIER DIAGNOSTICS

For each domain and search seed, we pool valid candidates with finite verifier predictions and offline TED labels, then compute MSE and Pearson r.