Supplementary Material: A Foundation Model for Zero-Shot Logical Rule Induction

A Synthetic Data Generation

Training a foundation model for rule induction requires diverse examples spanning the space of possible logical rules. Since real-world datasets are limited in quantity and coverage, we train exclusively on synthetically generated episodes. This section describes the data generation procedure in detail.

A.1 Episode Structure

Each training episode $\mathcal{E} = (X, Y, \mathcal{R}^*)$ consists of:

- $X \in \{0,1\}^{M \times (N+S)}$: Boolean feature matrix with M examples, N causal variables, and S spurious variables
- $Y \in \{0,1\}^M$: Binary labels
- $\mathcal{R}^*$: Ground-truth DNF rule defined over variables $\{1, \ldots, N\}$

The number of causal variables N and examples M are sampled from discrete uniform distributions for each episode: $N \sim \text{Unif}\{N_{\min}, \ldots, N_{\max}\}$ and $M \sim \text{Unif}\{M_{\min}, \ldots, M_{\max}\}$. In our experiments, $N \in \{6, \ldots, 12\}$ and $M \in \{24, \ldots, 48\}$.

A.2 DNF Rule Sampling

For each episode, we sample a random DNF rule $\mathcal{R}^* = C_1 \vee C_2 \vee \cdots \vee C_K$ where each clause $C_k \subseteq \{1, \ldots, N\} \times \{+, -\}$ is a set of signed literals. The sampling procedure is:

1. Sample the number of clauses $K \sim \text{Unif}\{1, \ldots, K_{\max}\}$
2. For each clause C_k:
   (a) Sample the clause length $L_k \sim \text{Unif}\{1, \ldots, \min(L_{\max}, N)\}$
   (b) Sample L_k distinct variable indices without replacement from $\{1, \ldots, N\}$
   (c) For each selected variable i, sample polarity $p_i \sim \text{Bernoulli}(0.5)$ to form literal (i, p_i)

This procedure ensures that clauses contain no duplicate variables and that literal polarities are balanced. We do not canonicalize rules (e.g., by removing subsumed clauses), accepting some redundancy in exchange for sampling simplicity. In our experiments, $K_{\max} = 6$ and $L_{\max} = 4$.

A.3 Example Generation

Given the sampled rule $\mathcal{R}^*$, we generate the causal feature matrix $X_{\text{causal}} \in \{0,1\}^{M \times N}$ and labels Y:

1. Sample each entry $X_{m,n} \sim \text{Bernoulli}(0.5)$ independently
2. Compute labels by evaluating the DNF rule:

$$Y_m = \mathcal{R}^*(X_m) = \bigvee_{k=1}^{K} \bigwedge_{(i,p) \in C_k} \ell_p(X_{m,i}) \qquad (1)$$

where $\ell_+(x) = x$ and $\ell_-(x) = 1 - x$

The uniform random sampling produces class imbalance that depends on rule structure. A single clause of length L yields $P(Y{=}1) = 2^{-L}$, so longer clauses produce fewer positives. Adding more clauses increases the positive rate via union, partially offset by clause overlap. This creates diverse class ratios across episodes.

A.4 Spurious Environment Features

A key challenge in rule induction is distinguishing *causal* features (those in the true rule) from *spurious* features (correlated with the label but not part of the rule). We augment each episode with S spurious variables that exhibit *environment-dependent* correlations.

The examples are conceptually split into two environments: Environment 1 (examples 1 to $\lfloor M/2 \rfloor$) and Environment 2 (examples $\lfloor M/2 \rfloor + 1$ to M). For each spurious feature s, we sample values with opposite correlations across environments. Let $\rho \in (0, 0.5)$ be the flip rate parameter:

- **Environment 1:** $P(s{=}1|Y{=}1) = \rho$, $P(s{=}1|Y{=}0) = 1 - \rho$
- **Environment 2:** $P(s{=}1|Y{=}1) = 1 - \rho$, $P(s{=}1|Y{=}0) = \rho$

After generating spurious features, we concatenate them to form $X = [X_{\text{causal}} \,|\, X_{\text{spurious}}] \in \{0,1\}^{M \times (N+S)}$, then randomly permute all examples to hide environment boundaries.

Marginal Independence by Design. An important property of this construction is that, marginally over the full dataset, each spurious feature is *independent* of the label:

$$P(s{=}1|Y{=}1) = \tfrac{1}{2}\rho + \tfrac{1}{2}(1-\rho) = \tfrac{1}{2} = P(s{=}1|Y{=}0) \quad (2)$$

This means simple correlation-based selection cannot distinguish spurious from causal features. However, spurious features exhibit a characteristic pattern: they "work" for half the examples and "fail" for the other half. The model must learn to detect this inconsistency through the statistical encoder's attention over examples, combined with the counterfactual necessity loss that penalizes selecting features whose removal does not change predictions.

In our experiments, we use $S = 3$ spurious features with flip rate $\rho = 0.3$.

A.5 Data Generation Parameters

Table 1 summarizes the data generation parameters used in our experiments.

Parameter	Value
Causal variables (N)	$\{6, \ldots, 12\}$
Examples per episode (M)	$\{24, \ldots, 48\}$
Max clauses ($K_{\max}$)	6
Max literals per clause ($L_{\max}$)	4
Spurious variables (S)	3
Spurious flip rate (ρ)	0.3

Table 1: Synthetic data generation parameters. All distributions are discrete uniform over the specified ranges.

Scope of Coverage. Our generator covers sparse DNF rules with bounded complexity ($K \leq 6$, $L \leq 4$) over moderate-sized variable sets ($N \leq 12$). This does not uniformly sample the space of all Boolean functions, many of which require exponentially many DNF clauses. Rather, it targets the sparse, interpretable rules that are the focus of rule induction research.

B Literal Feature Vector

For each literal l_j (where $j \in \{1, \ldots, 2N\}$ indexes both positive literals x_i and negations $\neg x_i$), we compute a feature vector $\phi_j \in \mathbb{R}^{18}$ containing the following statistics computed from the episode (X, Y):

The co-occurrence strength captures how a literal's truth value correlates with other literals. For a literal l_j, we compute:

$$c_{j,k} = \frac{1}{M} \sum_{m=1}^{M} (l_j^{(m)} - \bar{l}_j)(l_k^{(m)} - \bar{l}_k) \tag{3}$$

where $\bar{l}_j = \frac{1}{M} \sum_m l_j^{(m)}$ is the mean truth value. The aggregate co-occurrence strength is $\bar{c}_j = \frac{1}{2N-1} \sum_{k \neq j} |c_{j,k}|$.

The class-specific co-occurrence features ($\bar{c}^+$, $\bar{c}^-$, etc.) are computed analogously but restricted to positive or negative examples respectively. These help identify literals that participate in conjunctive patterns within a class.

Observation rates handle missing data: when a literal's value is unknown for an example, that example is excluded from the truth rate calculation but contributes to the observation rate statistic. The observation-rate features (obs$^+$, obs$^-$, obs) therefore directly quantify the effective sample size used to estimate each truth rate, allowing downstream layers to discount statistically thin literals. This is the mechanism behind the missing-data behavior described in Section **??**.

Feature	Description
$P(l_j{=}1 \mid y{=}1)$	Truth rate among positive examples
$P(l_j{=}0 \mid y{=}1)$	Complement of above
obs$^+$	Observation rate among positive examples
$P(l_j{=}1 \mid y{=}0)$	Truth rate among negative examples
$P(l_j{=}0 \mid y{=}0)$	Complement of above
obs$^-$	Observation rate among negative examples
$P(l_j{=}1)$	Marginal truth rate
$P(l_j{=}0)$	Complement of above
obs	Overall observation rate
$\mathcal{H}(l_j)$	Binary entropy: $-p \log p - (1{-}p) \log(1{-}p)$
sgn$_j$	Literal polarity: 1 for positive, 0 for negation
(reserved)	Zero-padded slot for future features
$\bar{c}_j$	Mean absolute co-occurrence strength
$\bar{c}^+_{\text{abs}}$	Mean |co-occurrence| among positive examples
$\bar{c}^+$	Mean co-occurrence among positive examples
$\bar{c}^-_{\text{abs}}$	Mean |co-occurrence| among negative examples
$\bar{c}^-$	Mean co-occurrence among negative examples
$\Delta\bar{c}$	Difference: $\bar{c}^+ - \bar{c}^-$

Table 2: Components of the literal feature vector ϕ_j. Co-occurrence is computed as the centered covariance between literal truth values.

C Auxiliary Terms in $\mathcal{L}_{\text{cf}}$

The deployed counterfactual objective adds two small-weight regularizers to $\mathcal{L}_{\text{nec}} + \mathcal{L}_{\text{spur}}$. Let $C_k^{(m)}$ be the clause truth value (Eq. **??**), $\mathcal{P} = \{m : y_m = 1\}$ the set of positives, and $r_k^{(m)} = \text{softmax}_k(C_k^{(m)})$ the responsibility weights from the necessity term.

Clause coverage overlap ($\lambda_o = 0.1$) penalizes pairs of clauses that simultaneously fire on the same positive example, discouraging redundant coverage:

$$\mathcal{L}_{\text{ovl}} = \frac{1}{|\mathcal{P}| \binom{K}{2}} \sum_{m \in \mathcal{P}} \sum_{k<k'} C_k^{(m)} C_{k'}^{(m)} \tag{4}$$

Counterfactual load balance ($\lambda_c = 0.01$) is the negative mean per-positive responsibility entropy, encouraging each positive to be explained by more than one clause within the CF objective:

$$\mathcal{L}_{\text{cf-bal}} = -\frac{1}{|\mathcal{P}|} \sum_{m \in \mathcal{P}} \left(-\sum_k r_k^{(m)} \log r_k^{(m)} \right) \tag{5}$$

D Computational Scaling

Figure 1 reports inference latency and peak memory as the schema size N and example count M vary. Latency is nearly constant (∼7.5ms) as M increases from 32 to 512. For N-scaling, latency grows sub-linearly (4.2ms→11.8ms for a 32× increase from N=16 to N=512), while memory scales as $O(N^2)$ due to attention over $2N$ literals. At N=512, inference completes in under 12ms with 593MB peak memory.

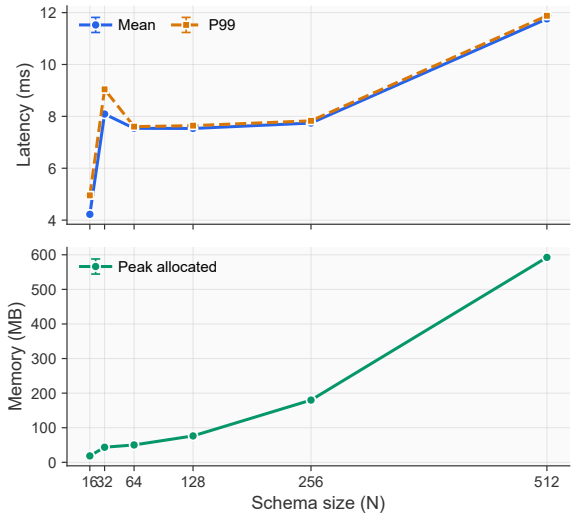

Figure 1: Computational scaling with problem size (N, left) and example count (M, right). Time remains nearly constant with M; memory scales quadratically with N due to attention.

E Baseline Descriptions

This section describes the baseline methods compared against our Neural Rule Inducer (NRI). The baselines span gradient boosting methods (non-interpretable accuracy ceilings), generalized additive models, and interpretable rule-based classifiers. All baselines are trained per-dataset with 5-fold cross-validation.

E.1 Non-Interpretable Ceilings

These methods provide strong accuracy baselines but do not produce human-readable rules.

XGBoost. XGBoost [Chen, 2016] is a highly optimized implementation of gradient boosted decision trees. It uses regularized learning objectives and efficient tree construction algorithms to achieve state-of-the-art performance on tabular data. The ensemble of trees is not directly interpretable, but provides a strong accuracy ceiling for comparison.

LightGBM (LGBM). LightGBM [Ke *et al.*, 2017] is a gradient boosting framework that uses histogram-based algorithms and leaf-wise tree growth for faster training and lower memory usage than traditional implementations. Like XGBoost, it serves as a non-interpretable accuracy ceiling.

E.2 Generalized Additive Models

Generalized additive models learn separate shape functions for each feature, providing partial interpretability through feature contribution graphs.

Explainable Boosting Machine (EBM). EBM [Lou *et al.*, 2013; Nori *et al.*, 2019] is a glassbox model that combines gradient boosting with generalized additive models. It learns a separate shape function for each feature, producing graphs showing how each feature contributes to predictions. While individual feature effects are interpretable, the model does not produce explicit logical rules and can include pairwise interaction terms that reduce transparency.

E.3 Rule-Based Classifiers

These methods produce explicit logical rules in various forms. Rule lists (RIPPER) evaluate rules sequentially; rule ensembles (RuleFit) weight rules linearly; tree-based methods (DT, FIGS) use hierarchical structures.

RIPPER. RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [Cohen and others, 1995] is a classic rule learning algorithm that grows rules greedily and then prunes them to optimize coverage and accuracy. It produces an ordered list of if-then rules that are evaluated sequentially. RIPPER is one of the most widely-used interpretable classifiers and serves as a primary baseline for rule induction.

RuleFit. RuleFit [Friedman and Popescu, 2008] extracts rules from an ensemble of decision trees and uses them as features in a sparse linear model. Each rule has an associated weight indicating its importance. While individual rules are interpretable, the weighted combination of many rules can reduce overall transparency compared to pure rule lists.

FIGS. FIGS (Fast Interpretable Greedy-tree Sums) [Tan *et al.*, 2025] learns a sum of small decision trees, each constrained to be shallow. The algorithm uses greedy fitting with early stopping to prevent overfitting. The resulting model is interpretable as a collection of small trees whose outputs are summed.

Decision Tree (DT). A standard CART-style decision tree [Loh, 2011] that recursively partitions the feature space using axis-aligned splits. We use scikit-learn's implementation with default hyperparameters. Decision trees are inherently interpretable as hierarchical rule structures, though the tree representation differs from DNF.

Decision Tree to DNF (DT-DNF). A decision tree converted to Disjunctive Normal Form by extracting the conjunction of conditions along each path from root to a positive leaf. Each path becomes a clause in the resulting DNF rule. This provides a direct comparison of tree-derived DNF rules against our neural approach.

E.4 Neural DNF Methods

Neural approaches to DNF learning use differentiable approximations of logical operations, enabling gradient-based optimization of rule structures.

Neural DNF (Scratch) (N-DNF). A neural network architecture designed to learn DNF rules, trained from scratch on each dataset. The architecture uses differentiable logic gates (sigmoid activations approximating AND/OR) similar to our approach, but without pre-training on synthetic data. This baseline tests whether per-dataset neural DNF learning outperforms our zero-shot transfer approach.

E.5 Summary

Table 3 summarizes the methods, their output types, and interpretability characteristics.

Table 3: Summary of baseline methods compared in our experiments.

Method	Type	Interp.	Training
XGB	Gradient Boosting	No	Per-dataset
LGBM	Gradient Boosting	No	Per-dataset
EBM	GAM	Partial	Per-dataset
RIPPER	Rule List	Yes	Per-dataset
RuleFit	Rule Ensemble	Partial	Per-dataset
FIGS	Tree Sum	Yes	Per-dataset
DT	Decision Tree	Yes	Per-dataset
DT-DNF	DNF via Tree	Yes	Per-dataset
N-DNF	Neural DNF	Yes	Per-dataset

F NRI Predicted Rules

This section presents example rules predicted by NRI on UCI datasets using 5% training data with auto-tuned clause selection. Rules are shown after removing duplicate clauses. Logical operators: $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT).

adult (1 clause)
($\neg$marital-status_Never-married $\wedge$ $\neg$relationship_Not-in-family $\wedge$ $\neg$relationship_Own-child $\wedge$ $\neg$sex_Female)

breast-cancer-wisconsin (1 clause)
(Clump_Thickness_gt_median $\wedge$ Cell_Size_Uniformity_gt_median $\wedge$ Cell_Shape_Uniformity_gt_median $\wedge$ Bare_Nuclei_gt_median)

car (2+1+3+1 clauses for 4 classes)

- **acc:** ($\neg$persons_2 $\wedge$ $\neg$lug_boot_big $\wedge$ $\neg$lug_boot_small $\wedge$ $\neg$safety_low) $\vee$ ($\neg$persons_2 $\wedge$ persons_more $\wedge$ $\neg$safety_low)
- **good:** ($\neg$maint_vhigh $\wedge$ $\neg$persons_2 $\wedge$ $\neg$lug_boot_small $\wedge$ $\neg$safety_low)
- **unacc:** (persons_2 $\wedge$ safety_low) $\vee$ (persons_2 $\wedge$ $\neg$safety_high) $\vee$ (persons_2)
- **vgood:** ($\neg$maint_vhigh $\wedge$ $\neg$persons_2 $\wedge$ $\neg$safety_low $\wedge$ $\neg$safety_med)

credit-approval (3 clauses)
($\neg$A11_gt_median $\wedge$ $\neg$A15_gt_median $\wedge$ $\neg$A7_h $\wedge$ $\neg$A10_t) $\vee$ ($\neg$A11_gt_median $\wedge$ $\neg$A15_gt_median $\wedge$ $\neg$A9_t $\wedge$ $\neg$A10_t) $\vee$ ($\neg$A11_gt_median $\wedge$ $\neg$A15_gt_median $\wedge$ $\neg$A6_x $\wedge$ $\neg$A10_t)

diabetes (1 clause)
(plas_gt_median $\wedge$ pres_gt_median $\wedge$ skin_gt_median $\wedge$ age_gt_median)

german-credit (3 clauses)
(checking_status_no checking $\wedge$ $\neg$purpose_domestic appliance $\wedge$ $\neg$employment_1$\leq$X$<$4 $\wedge$ $\neg$property_magnitude_no known property) $\vee$ (checking_status_no checking $\wedge$ $\neg$employment_1$\leq$X$<$4) $\vee$ (checking_status_no checking $\wedge$ $\neg$purpose_domestic appliance $\wedge$ $\neg$employment_1$\leq$X$<$4)

hepatitis (3 clauses)
($\neg$BILIRUBIN_gt_median $\wedge$ $\neg$ALK_PHOSPHATE_gt_median $\wedge$ $\neg$SGOT_gt_median) $\vee$ ($\neg$BILIRUBIN_gt_median $\wedge$ $\neg$ALK_PHOSPHATE_gt_median $\wedge$ $\neg$SGOT_gt_median $\wedge$ $\neg$MALAISE_yes) $\vee$ ($\neg$BILIRUBIN_gt_median $\wedge$ MALAISE_no $\wedge$ SPIDERS_no $\wedge$ ASCITES_no)

ionosphere (2 clauses)
(a01 $\wedge$ a21_gt_median $\wedge$ a25_gt_median $\wedge$ a33_gt_median) $\vee$ (a01 $\wedge$ a21_gt_median $\wedge$ a25_gt_median)

kr-vs-kp (6 clauses)
(bkxwp_f $\wedge$ $\neg$bkxwp_t $\wedge$ bxqsq_f $\wedge$ $\neg$bxqsq_t) $\vee$ ($\neg$bkxwp_t $\wedge$ bxqsq_f $\wedge$ $\neg$bxqsq_t $\wedge$ $\neg$wknck_t) $\vee$ (bkxwp_f $\wedge$ bxqsq_f $\wedge$ $\neg$bxqsq_t $\wedge$ wknck_f) $\vee$ ($\neg$bkxwp_t $\wedge$ $\neg$bxqsq_t $\wedge$ $\neg$rimmx_f $\wedge$ $\neg$wknck_t) $\vee$ (bkxwp_f $\wedge$ bxqsq_f $\wedge$ $\neg$bxqsq_t $\wedge$ wkna8_f) $\vee$ (bxqsq_f $\wedge$ $\neg$bxqsq_t $\wedge$ $\neg$rimmx_f)

mushroom (5 clauses)
($\neg$odor_n $\wedge$ $\neg$gill-spacing_w $\wedge$ $\neg$stalk-root_e $\wedge$ $\neg$spore-print-color_k) $\vee$ ($\neg$odor_n $\wedge$ $\neg$stalk-root_e $\wedge$ $\neg$stalk-surface-above-ring_s $\wedge$ $\neg$spore-print-color_k) $\vee$ ($\neg$odor_n $\wedge$ $\neg$stalk-surface-above-ring_s $\wedge$ $\neg$spore-print-color_k $\wedge$ $\neg$spore-print-color_n) $\vee$ ($\neg$odor_n $\wedge$ $\neg$gill-spacing_w $\wedge$ $\neg$spore-print-color_k $\wedge$ $\neg$spore-print-color_n) $\vee$ ($\neg$odor_n $\wedge$ $\neg$stalk-surface-above-ring_s $\wedge$ $\neg$ring-type_p $\wedge$ $\neg$spore-print-color_k)

nursery (1+2+0+1+1 clauses for 5 classes)

- **not_recom:** (health_not_recom $\wedge$ $\neg$health_priority $\wedge$ $\neg$health_recommended)
- **priority:** ($\neg$parents_great_pret $\wedge$ $\neg$has_nurs_very_crit $\wedge$ $\neg$health_not_recom) $\vee$ ($\neg$parents_great_pret $\wedge$ $\neg$has_nurs_very_crit $\wedge$ $\neg$health_not_recom $\wedge$ health_recommended)
- **recommend:** $\emptyset$ (empty rule)
- **spec_prior:** ($\neg$parents_usual $\wedge$ $\neg$has_nurs_less_proper $\wedge$ $\neg$has_nurs_proper $\wedge$ $\neg$health_not_recom)
- **very_recom:** ($\neg$parents_great_pret $\wedge$ $\neg$social_problematic $\wedge$ $\neg$health_not_recom $\wedge$ $\neg$health_priority)

spambase (1 clause)
($\neg$word_freq_hp_gt_median $\wedge$ $\neg$word_freq_hpl_gt_median $\wedge$ $\neg$word_freq_george_gt_median $\wedge$ $\neg$word_freq_meeting_gt_median)

tic-tac-toe (1 clause)
($\neg$middle-left-square_x $\wedge$ $\neg$middle-middle-square_o $\wedge$ middle-middle-square_x)

vote (1 clause)
($\neg$physician-fee-freeze_n $\wedge$ $\neg$education-spending_n $\wedge$ $\neg$crime_n $\wedge$ $\neg$duty-free-exports_y)

References

[Chen, 2016] Tianqi Chen. Xgboost: A scalable tree boosting system. 2016.

[Cohen and others, 1995] William W Cohen et al. Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning*, pages 115–123, 1995.

[Friedman and Popescu, 2008] Jerome H Friedman and Bogdan E Popescu. Predictive learning via rule ensembles. 2008.

[Ke *et al.*, 2017] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. volume 30, 2017.

[Loh, 2011] Wei-Yin Loh. *Classification and regression trees*, volume 1. Wiley Online Library, 2011.

[Lou *et al.*, 2013] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2013.

[Nori *et al.*, 2019] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

[Tan *et al.*, 2025] Yan Shuo Tan, Chandan Singh, Keyan Nasseri, Abhineet Agarwal, James Duncan, Omer Ronen, Matthew Epland, Aaron Kornblith, and Bin Yu. Fast interpretable greedy-tree sums. *Proceedings of the National Academy of Sciences*, 122(7):e2310151122, 2025.