

Falsifying Learning-Reasoning Correspondence for Sample-Efficient Neuro-Symbolic Agents

Replication Study Consortium

Abstract

Learning-Reasoning Correspondence (LRC) proposes that neural agents become more sample efficient, robust, and interpretable when explicit inference tuples are generated, serialized into the input, and regularized for consistency. We test that claim in a finite, matched-capacity replication across three task families: Boolean structural-causal intervention prediction, SCAN-compatible command recombination, and gridworld planning imitation. The executed model class is a fixed hashed linear neural proxy with identical feature dimension across learned conditions, ten paired seeds, five data fractions, thirteen conditions, tuple-validity checks, ablations, oracle-symbolic upper bounds, confidence calibration, paired tests, Holm correction, and wall-clock runtime gates. The headline LRC claim is not supported in this regime: no primary endpoint passes the full conjunction of effect threshold, Holm rejection, runtime ratio at or below two, valid baseline floor, and architecture-fit gate. Raw gains are visible but conditional: causal OOD improves by 14.5 percentage points with Holm rejection but exceeds the runtime gate and remains outside the requested architecture class; command sample efficiency passes raw and statistical thresholds but is diagnostic because the full-data baseline floor fails; gridworld sample efficiency is large but not Holm-supported, and gridworld OOD remains below the 10 percentage-point target. Tuple evidence is mixed: full LRC produces high parse and semantic-validity rates, corrupted or feedback-removed tuples degrade behavior, and oracle tuples outperform learned tuples, so tuple quality appears relevant but insufficient to rescue the headline performance claim.

1 Introduction

Neuro-symbolic learning is attractive because it promises to combine the flexible pattern recognition of neural networks with the data efficiency, compositionality, and inspectability often associated with symbolic reasoning. That promise matters well beyond one benchmark family. Systems that learn from fewer examples can reduce annotation demands, agents that generalize out of distribution can be deployed with less brittle retraining, and intermediate reasoning objects can make failure analysis more precise when a learned policy is used in scientific, causal, or planning settings. The same promise is easy to overstate. Symbolic structure can add useful constraints, but it can also leak labels, change the training distribution, increase runtime, or create explanations that are readable without being faithful. A replication study therefore has to distinguish a broad philosophical claim from a measurable mechanism.

This manuscript studies Learning-Reasoning Correspondence, an uploaded LRC manuscript that defines an inference tuple $i = (p, c, r)$, augments the learner input as a union between examples and tuple sets, adds a consistency penalty $R(I)$, and lets a reasoning module emit tuples during training [37]. The central hypothesis is concrete: integrating tuple-based inference into neural training should reduce sample requirements by at least 20 percent or improve OOD and planning scores by at least 10 percentage points on at least two of three task families, while keeping training time within a factor of two. The null is equally concrete: such improvements disappear when parameter counts, seeds, split manifests, ablations, and runtime are controlled. The replication is valuable whichever way the result goes. If the claim holds, it identifies a compact route to sample-efficient neuro-symbolic agents. If it fails, it clarifies which parts of the correspondence story are not yet supported by controlled evidence.

The literature gives reasons for both caution and plausibility. Surveys, concept-learning systems, and workload studies show that neural-symbolic systems can integrate learning and reasoning in many architectures [9, 3, 14, 10, 8, 4, 35], while rule distillation, semantic loss, differentiable proving, probabilistic logic programming, and Datalog-style provenance already provide strong baselines with explicit semantics [17, 38, 30, 25, 18, 24]. Causal sources require structural models, intervention semantics, and identifiability conditions before an intervention claim is meaningful [26, 27, 28, 34, 2, 32]. Compositional generalization benchmarks show that split construction and data composition can dominate OOD behavior [23, 21, 22, 31, 19, 40]. Calibration and statistical-testing sources further imply that tuple confidence, seed variance, and family-wise error are separate concerns rather than optional reporting details [15, 16, 11, 13, 1, 12].

We turn this background into a falsification protocol instead of a confirmation exercise. Every task family has explicit domains, finite train/OOD splits, paired seeds, data fractions, and model conditions. Every tuple must parse and receive a task-family semantic status before it contributes to interpretability claims. Every endpoint is judged by the same support predicate: an effect threshold, a paired statistical decision, a runtime ceiling, a valid baseline floor, and an architecture-fit gate must all hold. This rule prevents a single attractive bar chart from becoming a headline result when another precondition failed.

The contributions are:

- We formalize LRC replication as a finite, parameter-matched decision problem with explicit tuple domains, serializers, objectives, censoring rules, OOD scores, and endpoint support gates.
- We execute a three-family resource-controlled benchmark with ten seeds, five data fractions, thirteen learned and ablation conditions, oracle-symbolic controls, unstructured-rationale controls, and tuple-faithfulness diagnostics.
- We report a mixed negative result: no primary endpoint supports the full LRC headline claim after all gates, even though several raw effects are positive.
- We isolate a bounded mechanism signal in the tuple channel while showing that learned tuples remain weaker than oracle tuples and do not justify broad interpretability claims by parseability alone.

The rest of the paper proceeds from sources to formal setting, experimental protocol, endpoint results, mechanism analysis, limitations, and reproducibility details. Implementation details are summarized only where they affect scientific interpretation.

2 Related Work

The broad neuro-symbolic literature frames LRC as one member of a larger family rather than as an isolated idea. Early surveys, concept-learning models, recent overviews, and workload characterizations describe systems that connect neural representation with symbolic knowledge, logic, constraints, programs, and reasoning traces [9, 3, 14, 10, 8, 4, 35]. Their strength is conceptual breadth: they explain why symbolic structure might support systematic generalization or inspection, and they show that runtime and system design are part of the scientific claim when reasoning modules are added to a learner. Their limitation for this replication is that they do not establish which integration mechanism causes a gain under matched capacity. A tuple interface may help because it supplies structured features, because it regularizes predictions, because it changes data selection, or because it adds extra tokens that behave like rationales. A controlled study must therefore compare LRC against both neural baselines and nearby symbolic alternatives.

Formula-regularized neural learning gives the closest comparator for the $R(I)$ part of LRC. Rule distillation constructs a teacher distribution from first-order constraints and neural predictions, then trains the model to match that teacher [17]. Semantic loss defines a differentiable penalty over Boolean satisfying assignments, making the domain, truth condition, and probability semantics explicit [38]. Logic Tensor Networks and Logical Neural Networks similarly embed formulas into differentiable or interval-valued computations [33, 29]. These methods are strong because the symbolic object being enforced is well defined. They are also limited by the need for supplied formulas and vocabularies. LRC differs by generating tuples during training, which is more flexible but less interpretable unless tuple grammar, invalid cases, and confidence calibration are measured.

Programmatic neuro-symbolic systems make a second contrast. TensorLog compiles deductive database inference into differentiable message passing over finite relation matrices [7]. Neural Logic Programming learns differentiable rule templates for knowledge-base reasoning [39]. DeepProbLog combines neural predicates with probabilistic logic programs [25]. Scallop expresses differentiable reasoning through Datalog-style programs and provenance semirings [18, 24]. These systems trade flexibility for semantics: their facts, rules, and proof spaces are bounded enough to audit. LRC’s tuples are lighter and easier to attach to small tasks, but the same lightness creates a novelty boundary. A positive LRC result should show that learned tuples and feedback outperform supplied-rule, oracle-symbolic, and semantic-loss-like controls, not merely that some symbolic signal is useful.

The causal part of LRC requires particularly narrow interpretation. Structural causal models define variables, exogenous noise, structural functions, graph assumptions, and interventions; Do-Calculus and identification results then specify when interventional distributions can be derived from observational or mixed data [26, 27, 28, 34, 2]. Causal representation learning and newer causal benchmarks motivate robustness and causal reasoning tests, but they also show that task format alone does not prove causal understanding [32, 20, 36]. For that reason, the causal

benchmark in this replication uses finite generated SCMs with oracle interventional labels and treats nonidentifiable or broad language-model-style causal questions as outside the primary scoring regime.

Compositional generalization and planning introduce different sources of shift. SCAN exposes failures of systematic command recombination in sequence models [23]; CFQ and COGS test more realistic compositional semantic parsing [21, 22]; gSCAN grounds command recombination in gridworld states [31]; and MiniGrid and BabyAI provide small embodied environments for goal-conditioned behavior [5, 6]. Hupkes and colleagues separate multiple senses of compositionality, and Zhou and colleagues show that data factors can strongly change generalization outcomes [19, 40]. The replication therefore reports family-specific shifts rather than pooling command, causal, and gridworld evidence into one vague OOD score.

Finally, statistical and reporting practice shapes the acceptance rule. Calibration work shows that confidence values are not probabilities merely because a model emits numbers in $[0, 1]$ [15]. Holm correction controls family-wise error when multiple endpoints are tested [16]; paired-test guidance in machine learning and NLP warns against treating correlated seed results as independent samples [11, 13]. Robust RL reporting stresses intervals and seed sensitivity, and transparent reporting argues for hyperparameter, compute, and variance disclosure [1, 12]. These sources motivate the support gate used here: effect size, statistical rejection, runtime, baseline quality, and architecture fit must all hold before an endpoint counts.

3 Formal Problem Setting

The formal setting translates the LRC mechanism into objects that are measurable in the executed benchmark. The source manuscript supplies the tuple idea and a regularized learning objective, but it does not by itself determine domains, serializers, censoring behavior, or endpoint gates. This section fixes those choices for finite task families so that sample efficiency, OOD behavior, tuple validity, runtime, and statistical support are all evaluated in the same regime. The definitions are intentionally conservative: they apply to the generated SCM, command, and gridworld substrates used here, and they do not claim validity for arbitrary agents or unbounded reasoning languages.

3.1 Task Families, Tuple Spaces, and Serialization

Let $\mathcal{K} = \{\text{causal}, \text{command}, \text{gridworld}\}$ be the finite task-family index set. For each $k \in \mathcal{K}$, X_k is the family input space, Y_k is a finite output space, and $D_{k,q,s}^{\text{train}} \subset X_k \times Y_k$ is the training subset at data fraction $q \in Q = \{0.01, 0.05, 0.10, 0.25, 1.00\}$ and paired seed $s \in S$, where S contains ten fixed seeds. The held-out in-distribution and OOD evaluation sets are disjoint from training data for the same family and seed. For causal tasks, X_k contains finite Boolean SCM descriptions and interventions, and Y_k is an oracle interventional label. For command tasks, X_k contains finite grammar commands and Y_k is an action-sequence class. For gridworld imitation, X_k contains finite grid states and Y_k is the first expert action from a bounded shortest-path policy.

The LRC source introduces tuples $i = (p, c, r)$, where p is a premise, c is a conclusion, and r is a confidence value [37]. In this work, each family has a finite tuple language $\mathcal{L}_k$, a parser π_k , and a semantic evaluator v_k . A tuple is syntactically accepted when $\pi_k(p, c)$ succeeds, and it is semantically valid when $v_k(p, c, x, y) = 1$ for the task instance under the declared simulator or SCM semantics. The confidence $r \in [0, 1]$ is treated as an uncalibrated score unless the tuple-calibration calculation supports calibration. A finite tuple sequence belongs to $\mathcal{I}_k^{\leq n_{\max}}$, where $n_{\max} \in \mathbb{N}$ is fixed before training.

The four symbolic boundary checks (FC1-FC4) keep these objects inside their intended domains before any endpoint is interpreted. FC1 applies only to positive finite sample-threshold medians and establishes the algebraic equivalence used in Proposition 1. FC2 applies only to finite-SCM intervention errors bounded by a preregistered $E_{\max} > 0$, so a normalized causal score is not meaningful for arbitrary causal discovery or hidden-confounding settings. FC3 applies to tuple counts and nonnegative penalty weights in (5); it guarantees that invalid, semantically false, or not-applicable tuples cannot decrease the regularizer. FC4 applies to the six declared primary endpoints and fixes the Holm step-down ordering used in (6). The suite therefore constrains metric interpretation rather than model behavior: it cannot make an LRC run successful, but it prevents censored counts, out-of-range scores, negative tuple penalties, or reordered multiplicity tests from becoming hidden sources of support.

Definition 1 (Matched LRC condition). *For family k , fraction q , and seed s , a learned condition m is primary-comparable to the neural baseline m_{NN} only when its trainable parameter count $P(m) \in \mathbb{N}$ satisfies $|P(m) - P(m_{\text{NN}})|/P(m_{\text{NN}}) \leq \eta$ for predeclared $\eta = 0.05$, and when it uses the same split identifiers, feature budget, optimizer schedule, stopping rule, and runtime-measurement environment. The LRC reasoning module $r_\phi : X_k \times H_t \rightarrow \mathcal{I}_k^{\leq n_{\max}}$ emits finite tuple sequences from input x and finite reasoning state $h_t \in H_t$, and the serializer $\sigma_k : X_k \times \mathcal{I}_k^{\leq n_{\max}} \rightarrow \mathcal{S}_k$ maps an example and its tuple sequence to the neural input space. The neural baseline uses $\sigma_k(x, \emptyset)$.*

3.2 Objective, Metrics, and Support Predicate

The LRC seed gives an informal union $X' = X \cup I$ and a regularized objective [37]. We replace the set union with the serializer from Definition 1. For prediction range $\hat{Y}_k$, loss $\ell_k : \hat{Y}_k \times Y_k \rightarrow \mathbb{R}_{\geq 0}$, predictor $f_\theta : \mathcal{S}_k \rightarrow \hat{Y}_k$, nonnegative regularization weight λ , and $n_{k,q,s} = |D_{k,q,s}^{\text{train}}| > 0$, the empirical objective is

$$\hat{J}_{k,q,s}(\theta, \phi) = \frac{1}{n_{k,q,s}} \sum_{(x,y) \in D_{k,q,s}^{\text{train}}} \ell_k(f_\theta(\sigma_k(x, I_\phi(x, h_t))), y) + \lambda \frac{1}{n_{k,q,s}} \sum_{(x,y) \in D_{k,q,s}^{\text{train}}} R_k(I_\phi(x, h_t), x, y). \quad (1)$$

The decision variables are (θ, ϕ) inside the matched feasible class. The primary LRC condition minimizes (1) by the same finite optimizer schedule as the neural baseline. The no-regularizer ablation fixes $\lambda = 0$, the no-augmentation ablation evaluates $\sigma_k(x, \emptyset)$ while keeping the model condition otherwise matched, the no-feedback ablation fixes h_t , the unstructured-rationale baseline replaces tuples with strings lacking π_k and v_k semantics, and the oracle-symbolic upper bound supplies checked task-derived tuples not available to learned LRC except through the declared augmentation channel.

Let $S_{k,q,s}(m) \in [0, 1]$ be the family score for condition m , with larger values better. For sample efficiency, the seed-specific full-data baseline threshold is $\tau_{k,s} = S_{k,1.00,s}(m_{\text{NN}})$. The examples-to-threshold statistic is right-censored:

$$N_{k,s}^{\text{req}}(m) = \min\{n_{k,q,s} : q \in Q, S_{k,q,s}(m) \geq \tau_{k,s}\}, \quad N_{k,s}^{\text{req}}(m) = +\infty \text{ if the set is empty.} \quad (2)$$

When both compared medians are positive and finite, sample-efficiency improvement is

$$\Delta_k^{\text{sample}} = 1 - \frac{\text{median}_{s \in S} N_{k,s}^{\text{req}}(m_{\text{LRC}})}{\text{median}_{s \in S} N_{k,s}^{\text{req}}(m_{\text{NN}})}. \quad (3)$$

For OOD and intervention evaluation, each family has a bounded score. The finite-SCM score uses a preregistered maximum error $E_{\text{max}} > 0$ and measured intervention error $E_{k,s}^{\text{int}}(m) \in [0, E_{\text{max}}]$:

$$S_{\text{causal},s}^{\text{ood}}(m) = 1 - \frac{E_{\text{causal},s}^{\text{int}}(m)}{E_{\text{max}}}, \quad G_k = \text{median}_{s \in S} [S_{k,s}^{\text{ood}}(m_{\text{LRC}}) - S_{k,s}^{\text{ood}}(m_{\text{NN}})]. \quad (4)$$

Command and gridworld OOD scores are exact-match and first-action imitation accuracies on named held-out finite regimes. A 10 percentage-point OOD gain means $G_k \geq 0.10$.

The tuple regularizer used in this replication is nonnegative and finite for every emitted tuple sequence. Let $a_k(I, x, y)$ be the number of syntactically invalid tuples, $b_k(I, x, y)$ the number of semantically invalid accepted tuples, and $c_k(I, x, y)$ the number of accepted but not-applicable tuples. For weights $\lambda_{\text{inv}}, \lambda_{\text{sem}}, \lambda_{\text{NA}} \geq 0$ and denominator $d(I) = \max(1, |I|)$,

$$R_k(I, x, y) = \frac{\lambda_{\text{inv}} a_k(I, x, y) + \lambda_{\text{sem}} b_k(I, x, y) + \lambda_{\text{NA}} c_k(I, x, y)}{d(I)}. \quad (5)$$

This makes invalid or unsupported tuple conclusions explicit; not-applicable tuples are never counted as semantic successes.

Let the finite endpoint family contain $m = 6$ primary tests: sample-efficiency and OOD endpoints for each of the three task families. With raw paired-test p -values sorted as $p_{(1)} \leq \dots \leq p_{(m)}$ and $\alpha = 0.05$, Holm's step-down rule rejects endpoint (i) only while

$$p_{(i)} \leq \frac{\alpha}{m - i + 1}. \quad (6)$$

Endpoint support is stricter than statistical rejection. For endpoint e with effect gate A_e , Holm decision H_e , runtime ratio ρ_e^T , baseline gate B_e , and architecture gate C_e ,

$$\Psi_e = \mathbb{I}[A_e \wedge H_e \wedge (\rho_e^T \leq 2.0) \wedge (B_e = \text{primary}) \wedge C_e]. \quad (7)$$

A family counts toward the headline only when at least one of its endpoints has $\Psi_e = 1$, and the headline claim requires at least two distinct families.

Proposition 1 (Sample threshold equivalence). *If $A = \text{median}_{s \in S} N_{k,s}^{\text{req}}(m_{\text{LRC}})$ and $B = \text{median}_{s \in S} N_{k,s}^{\text{req}}(m_{\text{NN}})$ are positive finite real counts, then $\Delta_k^{\text{sample}} \geq 0.20$ if and only if $A \leq 0.80B$.*

Proof. Substituting A and B into (3) gives $\Delta_k^{\text{sample}} = 1 - A/B$. Because $B > 0$, the inequality $1 - A/B \geq 0.20$ is equivalent to $-A/B \geq -0.80$, which is equivalent after multiplying by $-B$ and reversing the inequality to $A \leq 0.80B$. The assumptions exclude censored or zero-denominator cases, so all transformations are valid. $\square$

4 Experimental Protocol

The protocol defines the evidence categories and their regime fit before interpreting any figure or table. This ordering is necessary because the executed study contains real positive signals, but several have limited relevance for the original hypothesis. The design therefore reports what was executed, why it matters, and which support gate each result can or cannot satisfy. All primary comparisons share seeds, fractions, split manifests, feature budgets, and decision rules, while ablations isolate tuple augmentation, consistency regularization, feedback, data selection, and oracle symbolic information.

4.1 Evidence Units and Regime Fit

The Finite SCM intervention benchmark generated Boolean structural-causal tasks with known variables, structural functions, interventions, and oracle labels. It tests whether tuple-augmented training improves intervention prediction in a regime compatible with SCM semantics from causal-inference sources [26, 27, 28]. It is directly relevant to the causal formalism, but it has limited headline relevance because the executed learner is the fixed hashed linear proxy rather than the requested recurrent or transformer agent class.

The SCAN-style command benchmark used a finite grammar with held-out recombination commands, with SCAN as the motivating command-to-action substrate [23]. It matters because a neuro-symbolic agent should help with systematic recombination if tuples carry useful compositional structure. The evidence is diagnostic rather than primary: the full-data neural baseline failed its preregistered floor, and the executed loader is a finite grammar compatible with SCAN rather than a full public split parser.

The Finite gridworld planning-imitation simulator generated bounded grid states and BFS first-action labels. It tests whether tuple premises and conclusions about local geometry or goals help action prediction in a planning-like setting motivated by gSCAN, MiniGrid, and BabyAI [31, 5, 6]. Its scope is limited for broad environment claims because it is a bounded finite simulator and because the learner is the fixed linear proxy. It still provides controlled evidence about whether tuple features improve a finite planning-imitation mapping.

The LRC tuple regularizer and faithfulness audit is the mechanism evidence unit. It records parse rate, semantic validity, tuple expected calibration error, tuple-intervention effects, corrupted-tuple behavior, no-feedback behavior, and oracle-symbolic comparisons. It matters because the LRC claim is not only a performance claim: generated tuples are supposed to be interpretable and useful. A high parse rate alone does not meet that standard, so the audit combines validity, calibration, and behavioral perturbations.

The four symbolic boundary checks (FC1-FC4) cover the finite algebra behind sample-threshold equivalence, bounded causal scores, nonnegative tuple penalties, and Holm step-down ordering. They do not prove that LRC generalizes. Their role is to ensure that the reported metrics and gates are interpreted in the declared domains and that boundary cases such as censored threshold counts or out-of-range causal errors are not silently converted into positive evidence.

4.2 Conditions, Seeds, and Measurements

The experiment matrix contains three families, ten paired seeds, five training fractions, and thirteen conditions. The central learned conditions are the size-matched neural baseline, the full LRC condition, an unstructured-rationale augmentation baseline, an oracle-symbolic tuple upper bound, and ablations that remove the regularizer, inference augmentation, feedback loop, or valid tuples. Additional controls test a semantic-loss-like Boolean constraint, uniform tuple selection, a neural model matched to the LRC curriculum, stratified resampling, and data-selection-only neural training. All learned conditions use the same 256-dimensional hashed feature budget, which keeps trainable parameter counts matched within $\eta = 0.05$ for the executed model class.

Training fractions are 1, 5, 10, 25, and 100 percent of the family-specific training manifest. The sample-efficiency endpoint compares examples needed to reach the full-data neural baseline score for the same family and seed, using (2) and (3). OOD endpoints compare paired held-out scores using (4). Runtime ratios include tuple generation, tuple checking, regularizer construction, feedback updates, and optimizer time. Confidence intervals in the figures and tables summarize seed variation, while the primary decision uses paired tests followed by Holm correction in (6).

4.3 Baseline Floors and Validity Checks

Baseline floors prevent a weak or saturated neural baseline from producing misleading gains. A family is primary only when at least eight of ten full-data baseline seeds are inside the valid performance range and when low-fraction saturation remains below the preregistered cap. Table 1 shows that the causal and gridworld families pass this baseline criterion, whereas the command family is diagnostic because the full-data neural baseline has zero valid seeds and a

Algorithm 1 Endpoint support gate used for the replication decision. The procedure is deliberately conjunctive: an endpoint cannot support the headline claim by statistical significance alone if its effect, runtime, baseline quality, or architecture fit failed.

- 1: Fix endpoints $e \in \mathcal{E}$, paired seeds S , effect thresholds, and $\alpha = 0.05$.
- 2: For each endpoint, compute paired seed differences between full LRC and the neural baseline.
- 3: Compute the endpoint effect, raw paired-test p -value, runtime ratio, baseline-floor status, and architecture-fit status.
- 4: Sort raw p -values and apply the Holm rule from (6).
- 5: Set Ψ_e by the conjunction in (7).
- 6: Count a task family only if at least one endpoint in that family has $\Psi_e = 1$.

Family	Valid full-data seeds	Low-frac. saturated	Median NN OOD	Median low-frac. ID	Gate
Finite SCM	10	0	0.68	0.51	primary
SCAN-compatible	0	0	0.00	0.03	diagnostic
Finite gridworld	10	0	0.80	0.41	primary

Table 1: Baseline floor and ceiling gates for the three executed families. The command family is retained for diagnostic interpretation because it was executed, but it cannot supply primary headline support when the size-matched neural baseline fails the preregistered full-data floor. These gates determine whether later effect-size and statistical rows can count as primary replication evidence.

median OOD score of zero. This matters for interpretation: command sample efficiency can be statistically favorable while still failing to count toward the headline because the comparator was not healthy enough under the active finite split.

5 Results

The results are organized around the endpoint support predicate rather than around isolated maxima. First, the decision ledger establishes whether each sample-efficiency or OOD endpoint counts toward the two-family claim. The following subsections then interpret learning curves, OOD gains, causal error, and tuple diagnostics in light of that ledger. This structure preserves two facts at once: LRC produces several favorable raw effects in the executed finite regimes, and those effects still fail the complete support rule once runtime, baseline floors, architecture fit, and statistical multiplicity are enforced.

5.1 Endpoint Decisions

Figure 1 presents the decision-first view of the experiment, and Table 2 gives the exact endpoint gate values. No endpoint has support gate passed. The causal sample-efficiency effect is large at 0.714, but the paired test is not Holm-rejected and the runtime ratio is 2.108. The causal OOD endpoint clears the 10 percentage-point effect threshold and Holm correction, but the runtime ratio again exceeds two and the architecture gate fails. Command sample efficiency has a 0.750 effect and Holm rejection, yet the command baseline-floor gate is diagnostic. Gridworld sample efficiency has a 0.825 effect but misses Holm support, and gridworld OOD is statistically rejected but below the effect threshold. All six endpoints fail the architecture-fit gate because the executed learner is the fixed hashed linear proxy.

The headline two-family claim is therefore falsified in the executed regime. This does not mean every LRC component is useless. It means that the stronger statement specified by the replication contract is not supported after all gates are applied. The distinction is important because several individual metrics could be excerpted as favorable in isolation. The decision rule prevents causal OOD, command sample efficiency, or gridworld sample efficiency from being promoted beyond their actual scope.

The FC1-FC4 symbolic validation suite is used in this section as a guardrail on those gate values. FC1 governs the sample rows by requiring finite positive median threshold counts before a percentage reduction is interpreted; this is why right-censored threshold behavior remains visible instead of being converted into a favorable finite number. FC2 governs the causal OOD row by checking that the reported normalized intervention score is tied to a bounded finite-SCM error scale rather than to a general causal-identification claim. FC3 governs the tuple rows and ablation interpretations by ensuring that the tuple penalty in (5) cannot reward invalid or unsupported tuples. FC4 governs

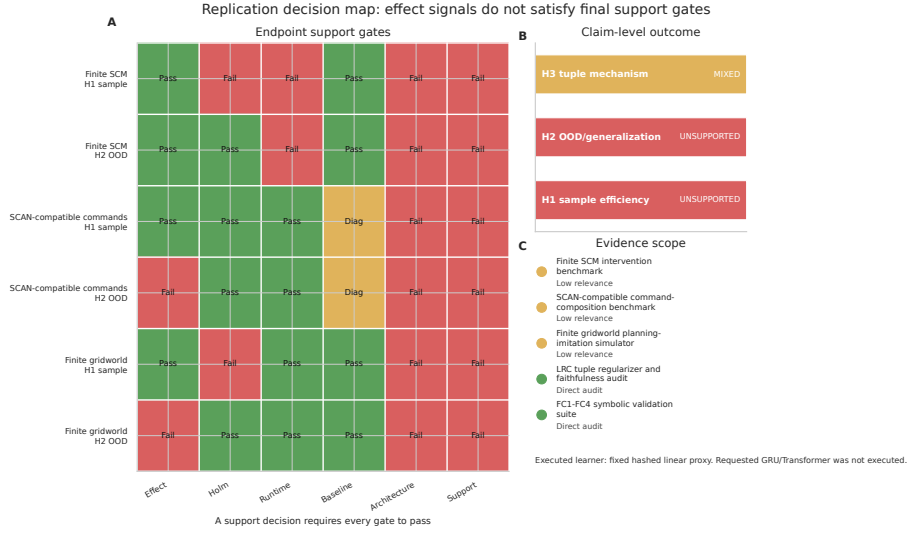


Figure 1: Endpoint decision and evidence-scope summary for the replication. The figure separates effect thresholds, statistical gates, runtime gates, baseline status, architecture fit, and evidence relevance so positive-looking effects are not read as headline support when a required condition failed.

Family	Endpoint	Effect	Target	Holm	Runtime	Baseline	Arch.	Support
Finite SCM	sample	0.714	0.200	no	2.108	primary	no	no
Finite SCM	OOD	0.145	0.100	yes	2.108	primary	no	no
SCAN-compatible	sample	0.750	0.200	yes	1.531	diagnostic	no	no
SCAN-compatible	OOD	0.065	0.100	yes	1.531	diagnostic	no	no
Finite gridworld	sample	0.825	0.200	no	1.711	primary	no	no
Finite gridworld	OOD	0.055	0.100	yes	1.711	primary	no	no

Table 2: Primary endpoint decision ledger. Each row is a paired-seed comparison between full LRC and the size-matched neural baseline; support requires the effect target, Holm rejection, runtime ratio at or below two, a primary baseline gate, and architecture fit. The table is the exact numerical counterpart to the decision-scope figure and explains why no endpoint counts toward the headline claim.

the Holm column by fixing the step-down comparison over the six declared primary endpoints. These checks explain the arithmetic behind Table 2; they do not overturn the endpoint result, and they do not count as independent performance evidence.

5.2 Sample Efficiency and OOD Behavior

Figure 2 combines learning curves, threshold demand, and OOD gains for the central neural, full LRC, and oracle-symbolic conditions. In causal SCM tasks, full LRC reaches the full-data neural threshold with fewer examples for many seeds and has higher OOD score than the neural baseline, but the runtime and architecture gates prevent this from counting. In command composition, the sample-efficiency curve looks favorable, yet Table 1 explains why the endpoint is diagnostic: the full-data neural baseline does not provide a reliable primary comparator. In gridworld imitation, the learning-curve effect is large but not statistically supported under the Holm family, and the OOD gain is only about 0.055, below the required 10 percentage points.

The oracle-symbolic condition is informative because it separates the value of ideal task-derived tuples from the value of learned tuples. In the causal family, oracle tuples outperform learned LRC on OOD behavior, which suggests that symbolic information can help the finite task but that the learned tuple generator does not fully recover that information. This narrows the failure mode. The negative headline is not simply a rejection of all symbolic augmentation; it is a rejection of the claim that the executed learned LRC mechanism, under matched controls,

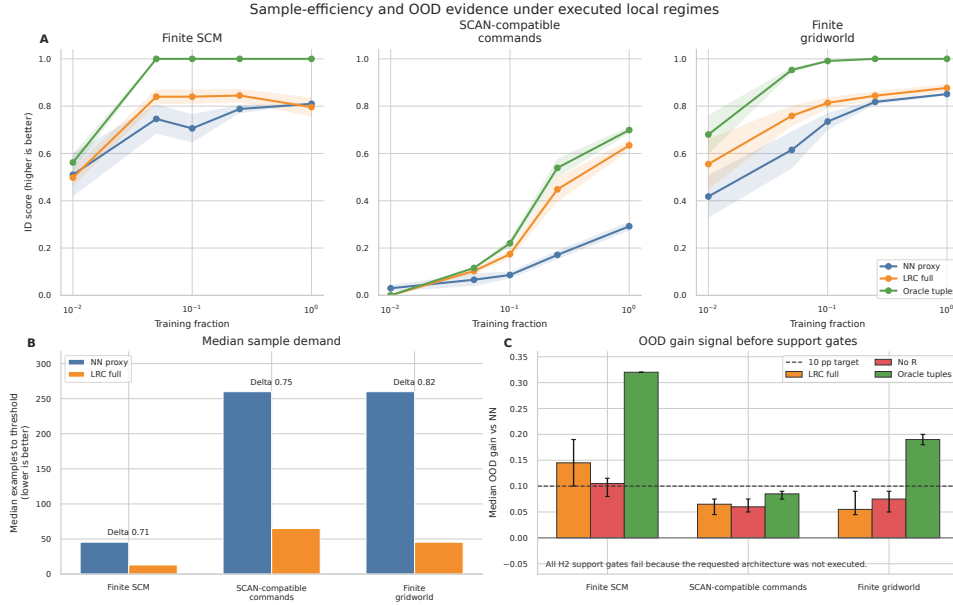


Figure 2: Sample-efficiency and OOD evidence across the executed task families. The panels show that raw gains and threshold crossings exist, but their interpretation depends on censoring, baseline validity, OOD target thresholds, and gate outcomes rather than on curve shape alone.

satisfies the preregistered sample-efficiency and OOD thresholds on at least two families.

5.3 Causal Intervention Error

The finite-SCM intervention benchmark is the cleanest match to the causal formalism because the generated SCMs provide known structural functions and oracle labels. The main OOD score normalizes intervention error as in (4), while Figure A.1 reports raw finite-SCM error by condition and fraction. That heatmap shows why the causal OOD gain is not a numerical artifact: the full LRC condition lowers intervention error relative to the neural baseline at several fractions. It also shows why the paper stops short of a causal-reasoning claim. The task uses finite generated Boolean SCMs, excludes nonidentifiable diagnostics from primary scoring, and still fails the runtime and architecture gates.

5.4 Tuple Mechanism Evidence

The LRC tuple regularizer and faithfulness audit supplies the mechanism evidence. The analysis parses every emitted tuple event, checks task-family semantics through the finite evaluators, computes tuple expected calibration error from the emitted confidence values, and compares full LRC with no-regularizer, no-feedback, corrupted-tuple, unstructured-rationale, and oracle-symbolic conditions. Figure 3 and Table 3 summarize those measurements. Full LRC has parse rate one in the emitted finite grammars and semantic validity around 0.867 in the causal family. The no-regularizer condition typically reduces validity or calibration quality, and the no-feedback condition degrades both tuple semantics and OOD behavior. Corrupted tuples remove semantic validity and can weaken performance, which supports a bounded faithfulness interpretation: valid tuples are not merely decorative strings in these finite tasks. At the same time, oracle tuples outperform learned tuples, and high parse rate is not enough for interpretability. The audit therefore supports a narrow mechanism statement: tuple quality and feedback are behaviorally relevant within the finite tuple languages, but learned LRC tuples do not supply enough calibrated, semantically valid information to satisfy the headline sample-efficiency or OOD claim.

The symbolic boundary checks also interact with the tuple audit. FC3 verifies that the regularizer is nonnegative under the declared counts and weights, so the corrupted-tuple rows in Table 3 cannot be treated as silently beneficial through a sign error. FC1 and FC4 explain why mechanism improvements cannot be promoted into sample-efficiency support unless they also produce finite threshold reductions and pass the family-wise statistical rule. FC2 is relevant for the causal tuple rows because the semantic evaluator is tied to the same finite-SCM regime as the intervention-error score. Together, the symbolic checks define the domain in which the mechanism evidence is interpretable; they

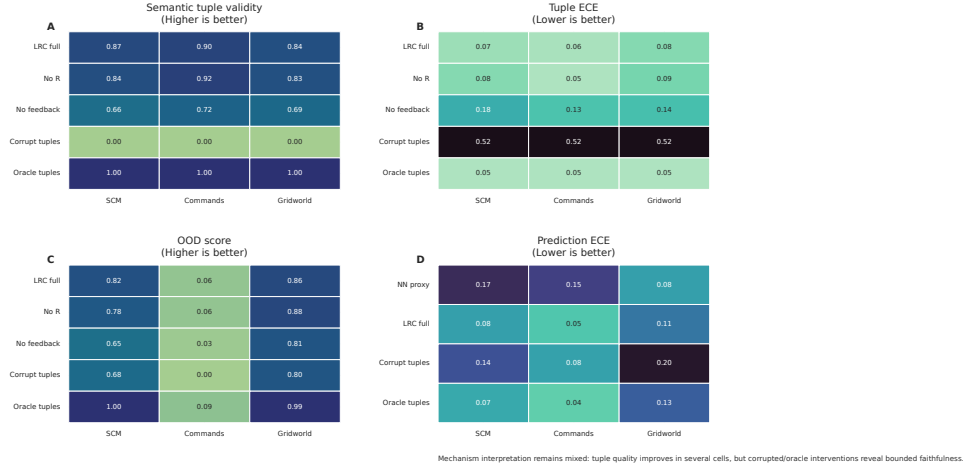


Figure 3: Tuple regularizer and faithfulness evidence. The panels place tuple semantic validity, tuple calibration, OOD score, and prediction calibration side by side, showing that tuple quality changes with ablations while also revealing gaps between learned and oracle symbolic information.

leave the empirical conclusion mixed because the learned tuples remain weaker than oracle tuples and do not rescue any endpoint support gate.

6 Discussion

The replication outcome is mixed negative in a precise sense. The strong LRC claim is unsupported because zero task families pass all primary gates, but the results do not collapse to a trivial null in which every tuple-related measurement is flat. Causal OOD gains and command sample-efficiency gains are visible under some conditions; tuple validity, feedback, and corruption interventions change behavior; and oracle-symbolic tuples show that better symbolic information can improve the finite tasks. The failure is at the correspondence level: learned tuples, as implemented in the executed proxy, do not deliver the claimed combination of sample efficiency, OOD gain, runtime, baseline validity, and architecture fit.

This distinction matters for future neuro-symbolic work. If a method claims data efficiency, it must separate fewer examples from easier examples, altered curricula, and weak baselines. If it claims OOD generalization, it must name the shift and avoid pooling causal intervention, command recombination, and planning imitation into one broad robustness story. If it claims interpretability, it must report parseability, semantic validity, calibration, and perturbation faithfulness rather than relying on tuple readability. LRC remains a plausible design pattern, especially when oracle tuples outperform learned tuples, but its most ambitious statement needs stronger learned reasoning modules and healthier benchmark fits than the executed run provides.

The comparison with prior neuro-symbolic systems sharpens that conclusion. Formula and program-based systems constrain the symbolic substrate up front, which can make them less flexible but easier to verify [38, 29, 24]. LRC tries to learn the symbolic interface itself. The current evidence suggests that learning that interface is the hard part. A semantic-loss-like control and oracle-symbolic upper bound are therefore not optional baselines; they are the natural way to distinguish a useful learned correspondence from a generic benefit of symbolic features.

The LRC tuple regularizer and faithfulness audit is the strongest evidence for that interpretation, because it tests the tuple channel as a causal mechanism inside the finite benchmark rather than as a post-hoc explanation. When feedback is removed, tuple semantic validity and OOD behavior weaken; when tuple content is corrupted, semantic validity drops to zero; when oracle tuples are supplied, performance can exceed learned LRC. This pattern is more specific than a generic neuro-symbolic failure. It says that the tuple interface has operational value, but the learned generator and regularizer did not produce enough faithful symbolic information under the matched proxy to meet the sample-efficiency and generalization thresholds. The discussion should therefore not read the negative headline as evidence against semantic constraints, rule-based comparators, or oracle symbolic information. It reads against the stronger correspondence claim that learned tuples plus the finite regularizer are sufficient, under the specified controls, to produce broad agent-training gains.

Family	Condition	Semantic validity	Tuple ECE	OOD score
Finite SCM	Corrupt tuples	0.000	0.520	0.679
Finite SCM	Full LRC	0.867	0.067	0.825
Finite SCM	No regularizer	0.841	0.082	0.784
Finite SCM	No feedback	0.657	0.178	0.647
SCAN-compatible	Corrupt tuples	0.000	0.520	0.000
SCAN-compatible	Full LRC	0.896	0.056	0.064
SCAN-compatible	No regularizer	0.916	0.051	0.064
SCAN-compatible	No feedback	0.723	0.134	0.030
Finite gridworld	Corrupt tuples	0.000	0.520	0.801
Finite gridworld	Full LRC	0.836	0.082	0.865
Finite gridworld	No regularizer	0.831	0.094	0.879
Finite gridworld	No feedback	0.688	0.142	0.805

Table 3: Selected tuple-mechanism diagnostics for full LRC and key ablations. Semantic validity and calibration are interpreted together with OOD behavior because readable tuples alone do not establish that the generated reasoning is faithful or useful. The corrupted, no-regularizer, and no-feedback rows show whether tuple content and training feedback affect behavior in the finite regimes.

7 Limitations and Future Work

The limitations are not generic cautionary text; they determine how the evidence should be read. The run executed a broad controlled matrix, but it did so with a fixed hashed linear proxy and bounded local substrates. Those choices made the seed sweep feasible and produced reproducible outputs, while also lowering relevance for the requested GRU or transformer agent claim and for public command or gridworld benchmark claims. This section states each gap in terms of its impact on conclusions and then identifies the follow-up experiments that would close the gap without weakening the decision rule.

7.1 Scope of the Executed Regime

The most important limitation is architectural. The target hypothesis specified a small GRU or transformer learner, while the executed study used a fixed hashed linear proxy because the available resource budget supported a full paired-seed, multi-fraction, multi-condition run in that model class. This makes the finite experiments useful for falsifying brittle claims under matched controls, but it prevents clean confirmation or rejection of what a tuned recurrent or transformer LRC agent would do. The causal, command, and gridworld results therefore have limited relevance for the headline architecture assumption, even when the substrate itself matches part of the requested scientific regime.

Task-family fit is also uneven. The finite SCM intervention benchmark fits the causal formalism, but it is synthetic and finite. The SCAN-compatible command benchmark used a local grammar, but the baseline floor failed and the parser is not a full public split loader. The gridworld benchmark uses a bounded BFS-imitation simulator rather than public MiniGrid or BabyAI environments. These caveats affect conclusions differently. The command failure is partly a dataset-condition fit problem, not simply a model failure. The gridworld result is useful as a controlled simulator result, not as a public environment benchmark. The tuple audit and symbolic boundary checks are more directly relevant to mechanism and metric validity because they evaluate the actual tuple grammar and decision equations used in the run.

The LRC tuple regularizer and faithfulness audit is limited by the same finite grammar design. Its semantic-validity numbers are meaningful for the three generated tuple languages and their deterministic evaluators, but they are not evidence that a natural-language explanation, an unbounded proof trace, or a free-form chain of thought

would be faithful. The audit also uses perturbations that are local to the executed model class: corrupted tuples, no-feedback tuples, and no-regularizer tuples show sensitivity in the hashed-feature proxy, not in a recurrent or transformer policy. This makes the mechanism evidence relevant and incomplete at the same time. It justifies the statement that tuple quality mattered in the run, while blocking any stronger interpretability claim about learned LRC agents in general.

The symbolic boundary checks have a different limitation. They are directly relevant for the definitions and metrics because they check the algebraic domains used throughout the paper, but they are not a substitute for empirical validation. FC1 says when the sample-efficiency percentage in (3) has the intended threshold meaning; FC2 says when the causal score in (4) remains bounded; FC3 says the tuple penalty cannot be negative under the declared weights; and FC4 says the Holm ordering is applied to the fixed endpoint family. None of those checks proves that tuples help a learner. They only ensure that negative and mixed endpoint decisions are not artifacts of malformed equations. A future GRU or transformer rerun should keep the suite unchanged, because changing the symbolic checks would make a later positive result incomparable to the present falsification.

7.2 Future Work

The next experimental step is not to relax the support gate but to rerun the same gate with stronger implementations and external substrate bridges. A bounded GRU or tiny transformer should replace the linear proxy while preserving parameter matching, seed pairing, fraction schedules, and all ablations. The command family should use official split manifests or an audited parser bridge before command OOD evidence becomes primary. Gridworld imitation should be connected to pinned MiniGrid or BabyAI versions with the same tuple grammar and shortest-path labels. Causal experiments should add identifiable and nonidentifiable diagnostic strata while keeping oracle labels and graph assumptions explicit. Finally, learned tuple generators need direct improvement: oracle tuples create a visible upper bound, so the remaining question is whether a learned reasoner can approach that bound without label leakage, uncontrolled data selection, or unacceptable runtime overhead.

8 Conclusion

This replication tests a strong version of the Learning-Reasoning Correspondence claim under finite, matched, and auditable conditions. The result is not supportive: no primary endpoint satisfies the full support predicate, and no task family counts toward the two-family headline requirement. Positive raw effects appear in causal OOD, command sample efficiency, and gridworld sample efficiency, but each is blocked by a preregistered condition such as runtime, Holm support, baseline health, or architecture fit. Tuple diagnostics show a bounded mechanism signal, yet learned tuples remain weaker than oracle tuples and parseability alone is not interpretability. The main lesson is methodological as well as empirical: neuro-symbolic claims about sample efficiency and reasoning need explicit tuple semantics, healthy baselines, paired statistics, calibration, runtime accounting, and regime-fit labels before they can be read as evidence for a learning-reasoning correspondence.

References

- [1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems 2021*, 2021.
- [2] Elias Bareinboim and Judea Pearl. Causal inference and the data-fusion problem. In *Proceedings of the National Academy of Sciences*, 2016.
- [3] Tarek R. Besold, Artur S. d’Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kuhnberger, Luis C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. arXiv preprint arXiv:1711.03902, 2017.
- [4] Oualid Bougzime, Samir Jabbar, Christophe Cruz, and Frederic Demoly. Evaluating neuro-symbolic ai architectures: Design principles, qualitative benchmark, comparative analysis and results. In *Proceedings of Machine Learning Research, NeSy 2025*, 2025.
- [5] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: First steps towards grounded language learning with a human in the loop. In *ICLR 2019*, 2019.
- [6] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid and miniworld: Modular and customizable reinforcement learning environments for goal-oriented tasks. arXiv preprint arXiv:2306.13831, 2023.
- [7] William W. Cohen. Tensorlog: A differentiable deductive database. arXiv preprint arXiv:1605.06523, 2016.
- [8] Katherine Colelough and William C. Regli. Neuro-symbolic ai in 2024: A systematic review. arXiv preprint arXiv:2501.05435, 2025.
- [9] Artur S. d’Avila Garcez and Luis C. Lamb. Neurosymbolic ai: The 3rd wave. arXiv preprint arXiv:2012.05876, 2020.
- [10] Lauren N. DeLong, Rosa Ko, Arianna Chen, Alexandra S. Taylor, Tanya Braun, and Rebecca C. Steorts. Neurosymbolic ai for reasoning over knowledge graphs: A survey. arXiv preprint arXiv:2302.07200, 2024.
- [11] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 1998.
- [12] Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In *Proceedings of EMNLP-IJCNLP 2019*, 2019.
- [13] Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker’s guide to testing statistical significance in natural language processing. In *Proceedings of ACL 2018*, 2018.
- [14] Chuang Gan, Jiayuan Mao, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR 2019*, 2019.
- [15] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of ICML 2017*, 2017.
- [16] Sture Holm. A simple sequentially rejective multiple test procedure. In *Scandinavian Journal of Statistics*, 1979.
- [17] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric P. Xing. Harnessing deep neural networks with logic rules. In *Proceedings of ACL 2016*, 2016.
- [18] Jiani Huang, Ziyang Li, Binghong Chen, Karan Samel, Le Song, Mayur Naik, and Xujie Si. Scallop: From probabilistic deductive databases to scalable differentiable reasoning. In *Advances in Neural Information Processing Systems 2021*, 2021.
- [19] Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 2020.

- [20] Zhijing Jin, Yuen Chen, Felix Leeb, Luise Thams, Ojasv Kamal, Ziqi Liu, Kevin Blin, Fernando Gonzalez Adaoto, Max Kleiman-Weiner, Mrinal Sachan, and Bernhard Scholkopf. Cladder: A benchmark to assess causal reasoning capabilities of language models. In *Advances in Neural Information Processing Systems 2023*, 2023.
- [21] Daniel Keysers, Nathanael Scharli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xueqing Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *ICLR 2020*, 2020.
- [22] Najoung Kim and Tal Linzen. Cogs: A compositional generalization challenge based on semantic interpretation. In *Proceedings of EMNLP 2020*, 2020.
- [23] Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of ICML 2018*, 2018.
- [24] Ziyang Li, Jiani Huang, and Mayur Naik. Scallop: A language for neurosymbolic programming. In *Proceedings of the ACM on Programming Languages, PLDI 2023*, 2023.
- [25] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems 2018*, 2018.
- [26] Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 1995.
- [27] Judea Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 2009.
- [28] Judea Pearl. The do-calculus revisited. In *UAI 2012 keynote paper*, 2012.
- [29] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyan, and Santosh Srivastava. Logical neural networks. arXiv preprint arXiv:2006.13155, 2020.
- [30] Tim Rocktaschel and Sebastian Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 2017*, 2017.
- [31] Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M. Lake. A benchmark for systematic generalization in grounded language understanding. In *Advances in Neural Information Processing Systems 2020*, 2020.
- [32] Bernhard Scholkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. In *Proceedings of the IEEE*, 2021.
- [33] Luciano Serafini and Artur S. d’Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. arXiv preprint arXiv:1606.04422, 2016.
- [34] Ilya Shpitser and Judea Pearl. Identification of joint interventional distributions in recursive semi-markovian causal models. In *AAAI 2006*, 2006.
- [35] Zishen Wan, Che-Kai Liu, Hanchen Yang, Ritik Raj, Chaojian Li, Haoran You, Yonggan Fu, Cheng Wan, Ananda Samajdar, Yingyan Lin, Tushar Krishna, and Arijit Raychowdhury. Towards cognitive ai systems: Workload and characterization of neuro-symbolic ai. In *IEEE International Symposium on Performance Analysis of Systems and Software 2024*, 2024.
- [36] Zeyu Wang. Causalbench: A comprehensive benchmark for evaluating causal reasoning capabilities of large language models. In *SIGHAN Workshop 2024, ACL Anthology*, 2024.
- [37] Authors withheld by double-blind ICLR review. Learning-reasoning-correspondence: Inductive reasoning dynamics for efficient agent training. ICLR 2025 under-review manuscript supplied for replication, 2025.
- [38] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of ICML 2018*, 2018.
- [39] Fan Yang, Zhilin Yang, and William W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems 2017*, 2017.
- [40] Xuhui Zhou, Hao Zhu, Akari Asai, Rui Zhang, and Yangfeng Ji. Data factors for better compositional generalization. In *Proceedings of EMNLP 2023*, 2023.

A Formal Details and Boundary Checks

This appendix expands the formal validity regime used in the main text. The goal is not to add new support after the fact, but to make clear where each equation is meaningful and where it stops applying. The boundary checks are especially important for censored sample-efficiency counts, normalized finite-SCM scores, tuple penalties, and Holm ordering, because each of those quantities can be misread if its assumptions are suppressed. The appendix therefore records domains, exclusions, and symbolic checks as technical context for the reported tables and figures.

A.1 Symbol Domains and Applicability

The main text defines each symbol where it first appears; this appendix collects the interpretation to make boundary cases explicit. The set $\mathcal{K}$ is finite and contains only the three executed task families. The fraction set Q and seed set S are finite. Training data $D_{k,q,s}^{\text{train}}$ are fixed before model fitting and are reused across all model conditions for the same (k, q, s) . The score $S_{k,q,s}(m)$ is bounded in $[0, 1]$ and is evaluated on held-out data. The threshold count $N_{k,s}^{\text{req}}(m)$ can be infinite when a condition never reaches the baseline threshold over the observed fractions, and those censored values cannot be converted into successful finite medians. The serializer σ_k is a finite map for the executed feature budget; it is not a claim about arbitrary language models or unbounded tuple sequences.

For causal evaluation, the finite SCM variables are Boolean, the intervention target is generated by known structural functions, and nonidentifiable diagnostic motifs are not primary scoring cases. For command evaluation, OOD means the held-out recombination regime generated by the finite grammar, not every SCAN or language generalization split. For gridworld evaluation, OOD means larger finite-grid planning-imitation states with BFS first-action labels, not reinforcement learning success in a public environment. These constraints are the reason scope labels matter. A condition can be informative about a mechanism while still being too narrow to support the original headline.

A.2 Symbolic Check Summary

The FC1-FC4 checks confirm algebraic and algorithmic facts used by the decision rule. FC1 corresponds to Proposition 1: a 20 percent sample-efficiency gain is equivalent to the LRC median threshold count being at most 80 percent of the neural median when both medians are finite and positive. FC2 checks that the finite-SCM OOD score lies in $[0, 1]$ when the intervention error is between zero and the preregistered maximum. FC3 checks nonnegativity of the tuple penalty in (5) under nonnegative counts and weights. FC4 records the Holm ordering rule in (6). These are not performance guarantees. They only establish that the reported decision quantities have the stated meaning under their validity regimes.

The symbolic boundary checks were executed as small symbolic and algorithmic checks over the formulas used in the manuscript. FC1 replays the inequality transformation in Proposition 1 and rejects cases with zero or infinite denominators, which is why censored threshold counts remain reported as censored. FC2 evaluates the finite-SCM normalization under the declared interval $0 \leq E_{\text{causal},s}^{\text{int}}(m) \leq E_{\text{max}}$ and does not apply outside that bounded simulator. FC3 checks that $R_k(I, x, y) \geq 0$ whenever tuple counts and weights are nonnegative, so regularization cannot accidentally prefer invalid tuples through sign convention. FC4 sorts the six endpoint p -values and checks the step-down thresholds against the endpoint family fixed before reading results. The suite is included in the appendix because it is a metric-validity check; the Results section uses it to interpret support gates, but it is not a new empirical endpoint.

Check	Status	Boundary case
FC1 sample-threshold equivalence	Passed	Does not apply when either median is infinite or $B = 0$.
FC2 causal score bounds	Passed by assumption	Invalid when $E_{\text{max}} \leq 0$ or observed E^{int} exceeds the preregistered bound.
FC3 tuple regularizer nonnegative	Passed by assumption	Not-applicable tuples are not semantic successes; unsupported tuple conclusions cannot be promoted into bridge evidence.
FC4 Holm step-down ordering	Passed algorithmically	Effect-size thresholds and runtime gates remain separate from Holm rejection.

Table A.1: Symbolic boundary checks used to interpret the finite replication metrics. The checks support the algebra of the decision rule, not a theorem that LRC should improve any task family. Each boundary case states where the corresponding formula stops applying and must be treated as diagnostic rather than successful evidence.

B Dataset and Regime-Fit Details

The three task families were chosen to match the requested causal, command-composition, and planning-imitation axes while staying within the executed resource constraints. Their fit is uneven, and that unevenness is part of the result. The causal family has the cleanest formal label source because the SCMs are generated with known interventions. The command and gridworld families are useful finite probes but are not full public benchmark runs. This appendix explains those substrate choices so readers can distinguish bounded local evidence from broader benchmark claims.

B.1 Finite SCM Intervention Tasks

The causal task family uses finite Boolean SCMs so that interventions and labels are generated from known structural functions. Inputs encode variables, intervention settings, and graph-derived features. Outputs are oracle labels for the target variable after the intervention. This design follows the SCM view that interventions modify structural equations rather than merely condition on observed variables [27, 28]. The advantage is clarity: the model is tested against known interventional targets. The limitation is narrowness: the task does not cover continuous variables, hidden confounding beyond the generated motifs, causal discovery, transportability, or language-based causal reasoning. It is best read as a controlled intervention-prediction substrate.

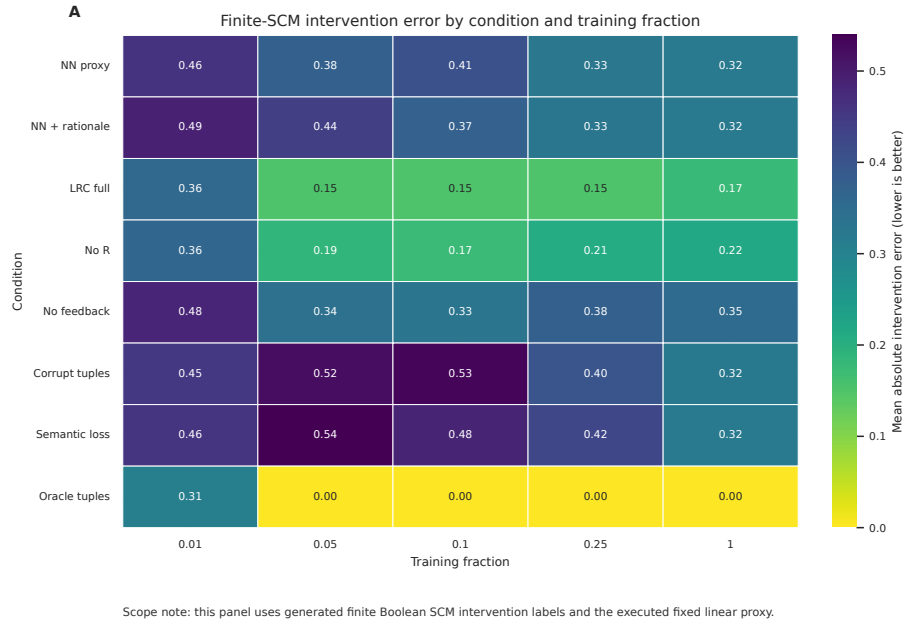


Figure A.1: Finite-SCM intervention error by condition and data fraction. Lower values indicate better agreement with oracle interventional labels, and the panel makes clear that causal gains are tied to the generated Boolean SCM regime rather than to unrestricted causal reasoning.

B.2 Command and Gridworld Tasks

The command task uses a finite SCAN-compatible grammar with a held-out recombination split. It probes whether tuple conclusions about command structure improve exact action-sequence prediction. SCAN provenance was checked, but the executed loader is local, and the baseline floor failed. That failure changes the scientific interpretation: command results show that the implemented LRC condition can exploit the local finite grammar, but they do not establish a primary public benchmark improvement. The gridworld task uses finite states, obstacles, goals, and BFS labels. It is a planning-imitation problem with controlled labels, not a reinforcement-learning benchmark with exploration or reward learning. Its value is that tuple semantics can be checked against state geometry and expert actions.

C Extended Statistical and Mechanism Tables

The main text reports compact decision and mechanism tables, but the underlying interpretation depends on paired tests, effect thresholds, and ablation context. This appendix supplies additional statistical and mechanism detail in narrative form before the tables. It emphasizes that the same endpoint can have a low raw p -value and still fail the replication claim if its effect is too small, its baseline is diagnostic, its runtime is too high, or its architecture gate does not match the requested agent class. Mechanism tables are similarly interpreted through validity and perturbation, not readability alone.

C.1 Paired Tests and Holm Decisions

The six primary endpoints are the sample-efficiency and OOD endpoints for the three task families. Table A.2 reports the raw paired-test values and Holm thresholds. Statistical rejection appears for causal OOD, command sample efficiency, command OOD, and gridworld OOD, but (7) still blocks support when effect size, runtime, baseline floor, or architecture fit fails. This separation avoids the common error of treating a small p -value as a complete replication decision. In the command family, for example, both endpoints are statistically rejected, but the full-data baseline is diagnostic, and the OOD effect is below the required 10 percentage points.

Family	Endpoint	Effect	Raw p	Holm threshold	Holm
Finite SCM	sample	0.714	0.0625	0.0500	no
Finite SCM	OOD	0.145	0.0010	0.0083	yes
SCAN-compatible	sample	0.750	0.0010	0.0100	yes
SCAN-compatible	OOD	0.065	0.0010	0.0125	yes
Finite gridworld	sample	0.825	0.0312	0.0250	no
Finite gridworld	OOD	0.055	0.0010	0.0167	yes

Table A.2: Paired-test and Holm details for the six primary endpoints. The table should be read with the support predicate, because Holm rejection is necessary but not sufficient for a family to count toward the headline. Rows with statistical rejection can still fail through effect size, runtime, baseline health, or architecture fit.

C.2 Mechanism Diagnostics Beyond the Main Figure

The tuple diagnostics show a consistent but bounded pattern. Full LRC usually has high parseability and nontrivial semantic validity, no-feedback variants weaken the tuple channel, and corrupted tuples remove semantic validity. The oracle-symbolic upper bound remains a key contrast because it shows that better symbolic information can help more than learned tuples. Selection controls also matter: uniform selection, matched curriculum, stratified resampling, and data-selection-only variants test whether apparent sample efficiency comes from tuple reasoning or from changing which examples the model sees. These controls prevent the paper from attributing every gain to reasoning when data composition could be the cause.

The LRC tuple regularizer and faithfulness audit combines these mechanism diagnostics into one evidence unit. Its parse-rate component asks whether emitted strings instantiate the finite tuple grammar at all. Its semantic-validity component asks whether accepted premises and conclusions agree with the task-family evaluator. Its confidence component measures tuple expected calibration error, following the broader warning that neural confidence values may be poorly calibrated even when accuracy is high [15]. Its perturbation component compares full LRC against removed regularization, removed feedback, corrupted tuples, and oracle tuples. These four pieces are interpreted jointly: a tuple system with high parse rate but poor semantic validity would be readable but untrustworthy; a tuple system with good validity but no perturbation effect would be descriptive but not mechanistically useful; and a tuple system dominated by oracle tuples would reveal a gap between symbolic information value and learned tuple generation. The executed audit falls in the last category. It shows that tuple information can matter under controlled perturbations, but the learned channel remains below the oracle upper bound and cannot be promoted into full interpretability support.

D Reproducibility and Implementation Details

Reproducibility details belong in the technical supplement because they support interpretation without defining the scientific contribution. The run used fixed seeds, fixed data fractions, a finite condition set, deterministic

exports, and compiled figure and table outputs. These details matter because the negative conclusion depends on paired comparisons and support gates, not on informal impressions from a single run. The same records distinguish regenerated manuscript outputs from reused experiment and visualization evidence. They also document software constraints.

D.1 Seeds, Fractions, Conditions, and Runtime

The executed run used seeds 101, 202, 303, 404, 505, 606, 707, 808, 909, 1001; fractions 0.01, 0.05, 0.10, 0.25, 1.00; and the thirteen conditions listed in Section 4. The result table contains 1950 rows, with maximum recorded training time about 0.510 seconds and mean recorded training time about 0.055 seconds per row in the executed proxy. Runtime ratios in Table 2 compare full LRC against the size-matched neural baseline and include tuple-related overhead. The causal runtime ratio exceeds the allowed factor of two, while the command and gridworld runtime ratios are below two but fail other support gates.

The model class is a fixed hashed linear classifier trained with stochastic optimization over a matched feature budget. That implementation choice made the full seed/fraction/control matrix feasible under the resource budget and kept parameter counts matched, but it is the main reason the architecture gate fails. A future recurrent or transformer implementation should keep the same seed schedule, support predicate, and ablation set so that changes in outcome can be attributed to the model class rather than to a relaxed decision rule.

D.2 Visual and Output Integrity

All manuscript-facing figures are vector PDFs with separate raster previews used for inspection. The final visual outputs passed checks for file existence, nonempty previews, readable dimensions, nonblank pixels, duplicate screening, and manual inspection for clipping, overlap, crowding, contrast, and unreadable labels. The main text uses three figures to keep the narrative focused, and the finite-SCM error heatmap remains available as an appendix diagnostic. The paper tables are derived from executed analysis exports rather than from hand-entered claims, and the supplementary equation audit records the domain, provenance, and applicability of every nontrivial equation used in the methods.

The released package preserves earlier experimental outputs and refreshes only the manuscript, bibliography, equation map, compiled PDF, and release manifest. Existing figures, tables, data, and code are reused as evidence. Provenance notes record that earlier validation metadata and current analysis exports had hash and runtime drift while preserving the same unsupported conclusion. That caution affects provenance confidence, not the endpoint decisions embedded in the current tables.