When Form Changes but Logic Doesn't: Building Logic-invariant LLMs through Structures

Xuyuan Liu[1] **Xinshuai Dong**[2] **Elynn Chen**[3] **Yujun Yan**[1]
[1]Dartmouth College [2]Carnegie Mellon University [3]New York University
{xuyuan.liu.gr, yujun.yan}@dartmouth.edu
xinshuad@andrew.cmu.edu elynn.chen@stern.nyu.edu

Abstract

Large language models (LLMs) have achieved strong performance across diverse reasoning tasks, yet it remains unclear whether this performance reflects an understanding of underlying logical structure or a reliance on shallow semantic cues. A key indicator of such understanding is whether models treat logically equivalent inputs consistently, producing the same predictions despite differences in expression. Through systematic evaluation on curated data, we show that even strong LLMs often fail this test: they do not consistently preserve predictions across logically equivalent inputs and generalize poorly to unseen logical forms. To address this limitation, we propose **LoGI**cal **ST**ructure-guided reasoning (LoGIST), a framework that guides LLM reasoning with explicit logical structure. LoGIST maps each input instance to a decision diagram, a directed acyclic graph that captures the underlying logical structure, and encodes this graph into a logical embedding that conditions the LLM's reasoning process. Across multiple training settings, LoGIST reduces inconsistency under logical equivalence and improves generalization to unseen logical forms, suggesting that logical invariance is a useful target for building more robust and generalizable logical reasoning in LLMs. [1]

1 Introduction

Large language models (LLMs) have demonstrated remarkable performance across a wide range of reasoning tasks, from mathematical problem to logical puzzles [56, 65, 55, 64, 58]. However, whether LLMs' reasoning capabilities reflect a genuine understanding of underlying logic or reliance on semantic cues remains unclear. Several recent studies suggest that, in some settings, LLMs may rely on surface-level patterns [28, 67, 63] and memorized world knowledge [61, 12, 3, 54, 25] rather than formal logical structures. A central test of logical understanding is whether models capture logical invariance: logically equivalent inputs should yield the same conclusions despite differences in expression. In this paper, we show that even state-of-the-art LLMs often fail this test. Using our curated dataset of logically equivalent input variants, we find that advanced LLMs exhibit substantial performance drops and frequent decision flips when the equivalent underlying logic is expressed in different forms. These failures reveal weak logical consistency and suggest that current LLMs remain overly sensitive to surface form rather than shared logical structure. We further show that addressing this invariance gap improves generalization to unseen logical forms.

Prior work on improving LLMs' logical reasoning has largely emphasized task-level accuracy over learning reusable logical structure. Some methods couple LLMs with external symbolic tools, such as logic solvers or theorem provers [33, 62, 31, 37]; while effective, they rely on external systems for formal reasoning, leaving the LLM's intrinsic structure-aware reasoning less examined. Other methods rely on data-driven training [39, 44, 29, 66], but do not explicitly model logical structure.

[1]Our code is available at https://github.com/GraphmindDartmouth/LoGIST.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

As a result, models may learn distribution-specific patterns rather than reusable logical principles, limiting generalization to unseen logical forms.

To address this gap, we propose **LoGI**cal **ST**ructure-guided reasoning (LoGIST). LoGIST maps each logical reasoning question to a decision diagram, a directed acyclic graph that captures the input's underlying logical structure independently of its surface form. Logically equivalent input variants are mapped to the same decision diagram. LoGIST then encodes these decision diagrams into structure-aware logical embeddings through contrastive learning and uses the embeddings to condition the LLM's reasoning. Across diverse training strategies, LoGIST improves both logical consistency across equivalent inputs and generalization to unseen logical forms, suggesting that logical invariance is a useful target for building more robust and generalizable logical reasoning in LLMs. Our further analysis shows that LoGIST learns to encode logical composition as a linear direction, thus explaining its generalizability.

In summary, our contributions are as follows:

- **New Insight**: We identify a key limitation of current LLMs: even strong models often fail to reason consistently over logically equivalent inputs, revealing weak logical invariance and limited generalization to unseen logical forms.
- **New Method**: We propose LoGIST, a framework that maps natural-language inputs to decision diagrams and encodes their logical structure into embeddings that condition LLM reasoning, enabling more consistent predictions across surface-form variants.
- **New Perspective**: We show that logical invariance and generalization are closely connected: explicitly modeling logical structure helps LLMs reason more consistently over equivalent inputs and generalize better to unseen logical forms.

2 Preliminaries

We begin by introducing the notation and key concepts for formal reasoning. We then discuss rules of replacement, which produce logically equivalent reformulations while preserving truth conditions. Finally, we present decision diagrams as representations that encode logical structure beyond surface-form differences.

We study formal reasoning tasks in which each instance consists of a premise set and a query. We denote each instance by $d = (\Gamma, \mathcal{Q})$, where $\Gamma = \mathcal{F} \cup \mathcal{R}$ is a finite premise set consisting of a fact set $\mathcal{F}$ and a rule set $\mathcal{R}$, and $\mathcal{Q}$ is the query whose truth value is to be determined under Γ. Each instance contains a paired formal and natural-language representation: $d = \big((\Gamma^{\mathrm{FL}}, \mathcal{Q}^{\mathrm{FL}}),\ (\Gamma^{\mathrm{NL}}, \mathcal{Q}^{\mathrm{NL}})\big)$, where $(\Gamma^{\mathrm{FL}}, \mathcal{Q}^{\mathrm{FL}})$ is the formal logical form and $(\Gamma^{\mathrm{NL}}, \mathcal{Q}^{\mathrm{NL}})$ is its natural-language realization, obtained through a verbalization function $\mathbb{T}$, such that $(\Gamma^{\mathrm{NL}}, \mathcal{Q}^{\mathrm{NL}}) = \mathbb{T}(\Gamma^{\mathrm{FL}}, \mathcal{Q}^{\mathrm{FL}})$. Each fact $f_i \in \mathcal{F}$ is a ground literal, *i.e.*, an atomic formula or its negation containing no variables, specifying a property of an entity or a relation among entities. For example, the fact *Carlos is good at soccer* is expressed as GoodAtSoccer(`Carlos`), where GoodAtSoccer$(\cdot)$ is a predicate and `Carlos` is a constant denoting a specific entity. Each rule $r_j \in \mathcal{R}$ is a first-order logical statement that specifies how one or more conditions imply a conclusion. Rules are built from predicates, logical connectives such as $\wedge, \vee, \neg, \rightarrow$, and, when applicable, quantifiers such as $\forall, \exists$. For example, the rule *if someone is good at soccer, then they join the soccer club* can be written as $\forall x(\,\text{GoodAtSoccer}(x) \rightarrow \text{JoinSoccerClub}(x))$, where x is a variable ranging over individuals. Given a consistent premise set Γ^{FL}, the task is to determine whether a query $\mathcal{Q}^{\mathrm{FL}}$ logically follows from the premises. The label is **True** if the premises necessarily imply the query ($\Gamma^{\mathrm{FL}} \models \mathcal{Q}^{\mathrm{FL}}$), **False** if they necessarily imply its negation ($\Gamma^{\mathrm{FL}} \models \neg\mathcal{Q}^{\mathrm{FL}}$), and **Uncertain** if neither conclusion follows. Here, $\models$ denotes classical logical entailment, meaning truth preservation from the premises to the conclusion.

Rule of Replacement The Rule of Replacement [42, 16, 14, 15] allows any subformula to be replaced by a logically equivalent one while preserving the truth value of the full formula. Examples include De Morgan's laws, commutativity, material implication, and other standard equivalence laws [9], which alter syntactic form while preserving logical meaning.

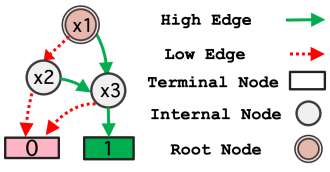


Figure 1: ROBDD example for function $F : (x_1 \vee x_2) \wedge x_3$ with variable order $x_1 \prec x_2 \prec x_3$.

Reduced Ordered Binary Decision Diagram (ROBDD) An ROBDD [5, 6] is a rooted directed acyclic graph representing

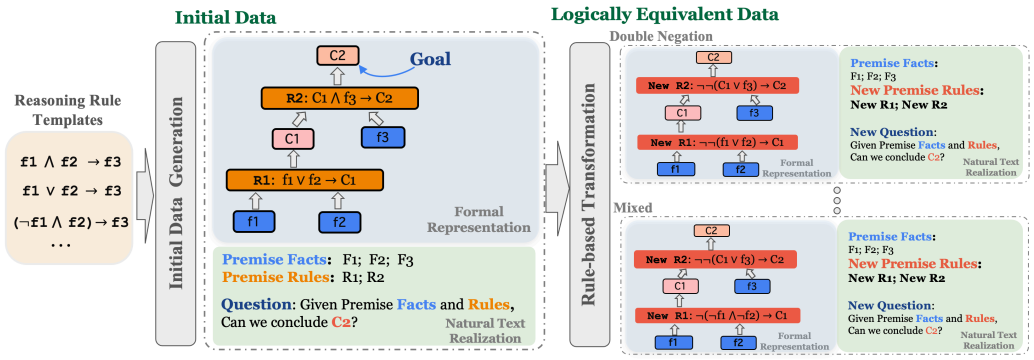


Figure 2: Pipeline for generating logically equivalent data. We instantiate reasoning problems from rule templates and apply rules of replacement to the premise rules while keeping the fact set and query fixed. This produces surface-form variants that are logically equivalent and share the same ground-truth label.

a Boolean function. Let $F : \{0,1\}^m \to \{0,1\}$ be a Boolean function defined over an ordered variable list $(x_1, \ldots, x_m)$, where each variable x_i corresponds to the truth value of a ground atom. An ROBDD contains a root node ρ, two terminal nodes τ_0 and τ_1 labeled with outputs 0 and 1, and internal nodes associated with Boolean variables. Each internal node labeled x_i has a low branch for $x_i = 0$ and a high branch for $x_i = 1$. We denote the sets of low- and high-branch edges by E_0 and E_1, respectively. Given a Boolean function F and a fixed variable order, an ordered BDD recursively branches on variables in that order until the function value is determined. The ROBDD is obtained by merging isomorphic subgraphs and removing nodes whose low and high successors are identical. Under a fixed variable ordering, logically equivalent formulas over the same variables induce the same ROBDD, yielding a canonical graph representation. We present an example ROBDD for the function $F : (x_1 \vee x_2) \wedge x_3$ with variable order $x_1 \prec x_2 \prec x_3$ in Fig. 1.

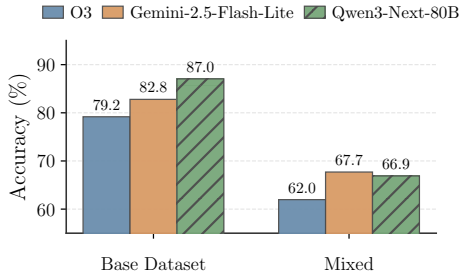
3 Challenge: LLMs Struggle with Logic-Invariant Reasoning

It remains unclear whether current LLMs base their predictions on the logical structure of the input, *i.e.*, the rules and dependencies that determine the query's truth value, or on surface-level cues such as entities, relations, and wording. To systematically examine this question, we construct a dataset by transforming each instance into multiple logically equivalent variants with different surface expressions. The full data generation pipeline is illustrated in Fig. 2. We then evaluate whether LLMs make invariant predictions across these variants.

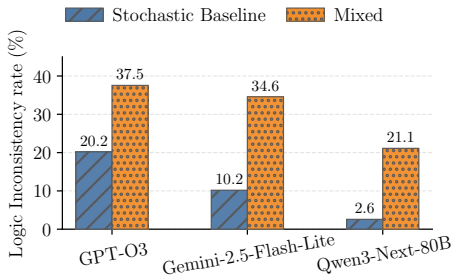
3.1 Constructing Logically Equivalent Data

Data Initialization We start by constructing a reasoning dataset in which each example contains both a formal representation and its corresponding textual realization. Building on ProverGen [39], we define a step-by-step deductive process where each step applies a grounded implicational rule to derive a new fact. The derived fact can either serve as a premise for subsequent reasoning steps or constitute the final answer. Using this procedure, we build the base dataset $\mathbb{D}_{\text{base}}$. Each example $d_i \in \mathbb{D}_{\text{base}}$ contains a pair of formal premises and query, $(\Gamma_i^{\text{FL}}, \mathcal{Q}_i^{\text{FL}})$, together with its corresponding natural-language realization, $(\Gamma_i^{\text{NL}}, \mathcal{Q}_i^{\text{NL}})$, which serves as the input to LLMs for reasoning.

Rule-Based Transformation under Logical Equivalence From $\mathbb{D}_{\text{base}}$, we construct logically equivalent variants by applying *rules of replacement* to the formal rule sets while keeping the fact sets fixed. Specifically, for each example $d_i \in \mathbb{D}_{\text{base}}$, we apply a transformation ϕ to its rule set $\mathcal{R}_i$, yielding $\Gamma_{i,\text{new}}^{\text{FL}} = (\mathcal{F}_i, \phi(\mathcal{R}_i))$. This transformation preserves the original facts and deductive dependency while changing the expression of the rules. We consider three replacement rules: *material implication*, *double negation introduction*, and *De Morgan's laws* (details in Tab. 7). Each transformed formal representation is then rendered back into natural language using the same procedure as in the base dataset construction. This process yields three transformed datasets, $\mathbb{D}_{\text{MI}}$, $\mathbb{D}_{\text{DNI}}$, and $\mathbb{D}_{\text{DM}}$, corresponding to material implication, double negation introduction, and De Morgan's laws, respectively. Each transformed dataset is paired example-wise with $\mathbb{D}_{\text{base}}$. We further construct



(a) Accuracy across logically equivalent datasets.



(b) Logic Inconsistency rate across logically equivalent datasets.

Figure 3: Accuracy (a) and Logical Inconsistency rates (b) of LLMs under logically equivalent transformations. Even advanced models exhibit substantial performance drops and instability on the transformed inputs, revealing a fundamental limitation in capturing the underlying logical structure.

a mixed dataset, $\mathbb{D}_{\text{Mixed}}$, by randomly sampling a replacement rule at each reasoning step. Data examples are provided in App. A.8.

3.2 Evaluating LLMs on Logically Equivalent Data

We evaluate LLMs' capability for logic-invariant reasoning using our curated dataset, and identify two key observations: **(i)** LLMs fail to reliably capture logically equivalent structures, and **(ii)** prompting with explicit rules does not mitigate this failure. Specifically, we test models on $\mathbb{D}_{\text{base}}$ and its logically equivalent transformations $\mathbb{D}_{\text{Mixed}}$ using two metrics: **Accuracy**, which measures correctness, and **Logic Inconsistency**, which measures the proportion of instances whose predicted label changes between the original input and its logically equivalent reformulation.

Observation 1: LLMs fail to capture logically equivalent structures. We evaluate closed-source models GPT-O3 [32] and Gemini-2.5-Flash-Lite [48], along with the open-source Qwen3-Next-80B [50], under a consistent protocol. Results are shown in Fig. 3a. Although these models perform well in the original setting, their accuracy consistently drops in the mixed setting, where reasoning requires integrating equivalent reformulations across multiple steps. The drop reaches up to 20%, suggesting that the models struggle to reason consistently across logically equivalent forms. We next evaluate logical inconsistency. On $\mathbb{D}_{\text{Mixed}}$, the evaluated models show substantial disagreement between original inputs and their logically equivalent variants, with disagreement rates reaching nearly 30%. To contextualize this disagreement, we compare it against a Stochastic Baseline, which measures run-to-run disagreement on the original evaluation set and estimates variation due to decoding randomness alone. Reformulation-induced disagreement exceeds this baseline by approximately 20% across all models, indicating that the inconsistency primarily reflects sensitivity to logically equivalent reformulations rather than stochastic decoding variation.

Observation 2: Prompting Is Insufficient to Induce Logic-Invariant Reasoning We further examine whether explicit prompting improves LLM performance on logically equivalent data. In this setup, each model is prompted with a set of replacement rules, presented through natural-language explanations and corresponding formulas, as shown in Fig. 8. We evaluate the model under the same protocol as above, with results reported in Tab. 1. In-context prompting yields inconsistent and limited gains in the mixed reasoning setting, leaving a substantial performance gap from the original dataset. This indicates that merely presenting replacement rules in the prompt is insufficient for LLMs to reliably recognize or apply logical equivalence.

Table 1: Mixed reasoning performance (by %): explicitly prompting LLMs to consider logical equivalence yields little to no improvement.

Model	Mixed
`Gemini-2.5-flash-lite`	67.7
`Gemini-2.5-flash-lite(w.Prompt)`	70.4
`Qwen3-Next-80B`	66.9
`Qwen3-Next-80B(w.Prompt)`	66.1

Logic structure as primary factor. Finally, we extend our study to illustrate that the observed failures and inconsistency reflect an inability to recognize logical equivalence rather than other factors,

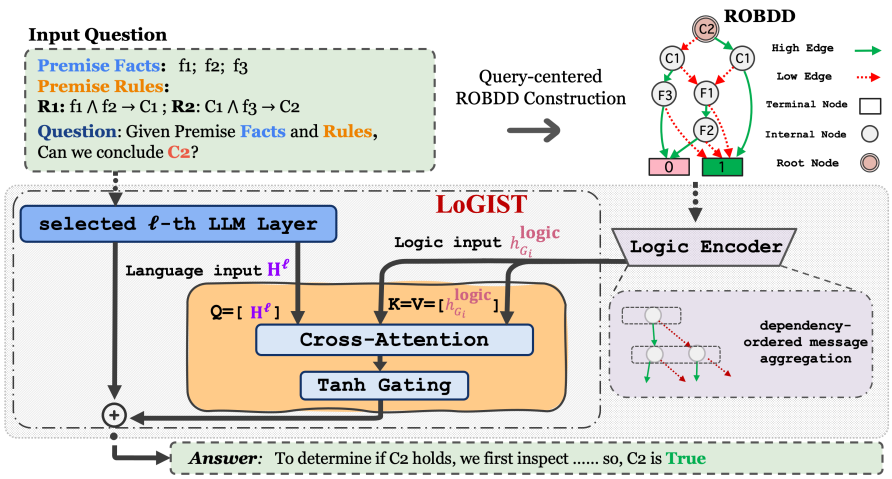


Figure 4: Framework of LoGIST. LoGIST maps each question to an ROBDD that captures its logical dependency structure, then encodes the ROBDD into a logic embedding that conditions selected LLM layers via gated cross-attention. This lets the LLM attend to explicit logical-structure information while reasoning.

e.g. differences in task difficulty after logic transformation. To achieve this, we first compare the Mixed variant with other logically equivalent variants that also involve complex rewrites, and we find differences of 19.2% between Mixed and Mat.Impl. and 18.3% between Mixed and Double Neg. Both significantly exceed the stochastic noise level of 2.6%, showing that this trend persists even for data of comparable difficulty. The flip-direction analysis further supports this interpretation: if increased difficulty introduced by logical transformations drove these flips, correct-to-incorrect transitions would dominate as difficulty increased. However, in practice, we find that the transitions are nearly symmetric (GPT-O3: 12.5% correct-to-incorrect versus 10.8% incorrect-to-correct), with similar trends observed for other models. These findings further suggest that difficulty is not the primary driver of decision flips. Together, these findings suggest that the primary reason for these failures is that LLMs struggle to recognize logical equivalence.

4 Method: Logical Structure-guided Reasoning

Building on our findings that current LLMs struggle with logic-invariant reasoning and that prompting alone does not reliably mitigate this issue, we turn to training-based methods, proposing LoGIST, which incorporates formal logical structure as auxiliary information to guide LLM reasoning.

4.1 Representing Logical Structure with ROBDDs

To capture logical equivalence, we augment each natural-language input with an explicit logical-structure representation. This representation should satisfy two properties: **(i)** it is grounded in truth-values, so that logically equivalent expressions share the same representation, and **(ii)** it captures the dependencies between grounded facts and rules. ROBDDs [5, 6] satisfy both: under finite grounding and a fixed variable ordering, logically equivalent formulations over the same grounded atoms induce the same ROBDD, and the variable ordering itself encodes the reasoning dependencies. To construct the ROBDD for an instance, we determine its variable ordering by starting from the target query and recursively collecting the facts and rules required to evaluate its truth value. We then build the diagram using this ordering. We present the detailed procedure for this step in Alg. 1.

4.2 Learning Logical Embeddings

Based on the constructed logic graph, our logic encoder aims to map each ROBDD to a graph-level representation while preserving its decision structure. Since an ROBDD represents a Boolean function through root-to-terminal decisions in a fixed order, the encoder in LoGIST should capture the ordered topology of internal nodes, distinguish between low and high branches, and preserve the Boolean values of terminal nodes. To this end, we design the encoder with partial-order-aware node updates,

edge-specific propagation for the two branch types, and a terminal-based readout that yields the final graph-level representation.

Concretely, for an ROBDD G_i, each internal node is initialized by averaging the token embeddings of its associated facts. Node states are then updated in topological order so that each node is processed after all its predecessors have been updated. Let $\boldsymbol{h}_v$ denote the current hidden state of node v. Messages arriving through low and high edges are aggregated separately. For edge type $k \in \{0, 1\}$, corresponding to low and high branches, respectively, the aggregated incoming message is $m_k(v) = \sum_{(u,v)\in E_k(G_i)} \sigma(\mathbf{W}_g^k \boldsymbol{h}_u) \odot (\mathbf{W}_m^k \boldsymbol{h}_u)$, where $E_k(G_i)$ denotes the set of edges of type k, and $\mathbf{W}_g^k$ and $\mathbf{W}_m^k$ are trainable edge-specific projections. Here, $\sigma(\cdot)$ is the sigmoid function, and $\odot$ denotes element-wise multiplication. The sigmoid term acts as a dimension-wise gate that controls how much of the projected message $\mathbf{W}_m^k \boldsymbol{h}_u$ is passed from u to v. The two aggregated messages are then summed and combined with the current state through a GRU: $\boldsymbol{h}'_v = \mathrm{GRU}(m_0(v) + m_1(v), \boldsymbol{h}_v)$., which is used for subsequent propagation. This propagation continues until the terminal nodes are updated, and the graph-level summary is read out from the two terminal states: $\mathbf{h}_{G_i} = \mathrm{MLP}([\boldsymbol{h}_{\tau_0} \| \boldsymbol{h}_{\tau_1}]) \in \mathbb{R}^{d_g}$. The full encoding procedure is detailed in Alg. 2.

Given this architecture, we train the encoder to learn representations that reflect ROBDD decision structure. To this end, we use contrastive learning [53, 41], which pulls structurally equivalent ROBDDs closer together in the embedding space while pushing structurally distinct ones apart. To construct these pairs, we assign each ROBDD a structure-only signature using bottom-up hashing [4, 2, 23]. Each node is assigned a hash value based on the hash values of its children, with the root's hash value serving as the signature of the entire diagram. This signature depends solely on structural information, regardless of the original semantic content of the associated facts. ROBDDs sharing the same signature form positive pairs, while those with different signatures but similar sizes form hard negative pairs. The encoder is trained on these pairs using the InfoNCE loss [52].

4.3 Guiding LLM Reasoning with Logical Structure

While the contrastively trained encoder produces structure-aware embeddings, these embeddings lie in a representation space separate from that of the LLM, and directly incorporating them into the LLM's hidden representations can be brittle. We therefore use a two-stage strategy: *logic-text alignment* maps logical embeddings into the LLM's representation space, and *logic fusion* uses the aligned embeddings as auxiliary features to guide reasoning.

To align logical embeddings with the LLM representation space, we train a lightweight aligner f_{align} while keeping both the logic encoder and the LLM frozen. Given the graph embedding $\mathbf{h}_{G_i}$, the aligner projects it into the LLM hidden dimension: $\mathbf{h}_{G_i}^{\mathrm{logic}} = f_{\mathrm{align}}(\mathbf{h}_{G_i}) \in \mathbb{R}^{d_t}$, where d_t is the LLM hidden dimension. We use the frozen LLM to obtain a paired text embedding $\mathbf{t}_{G_i} \in \mathbb{R}^{d_t}$ for each input by mean-pooling its hidden states. We also compute the mean $\boldsymbol{\mu}$ and top-r PCA directions $\mathbf{U} \in \mathbb{R}^{r\times d_t}$ from all training text embeddings to characterize the dominant text-representation subspace. The aligner is optimized with two complementary losses. A global subspace loss keeps aligned logical embeddings within the dominant LLM text-embedding subspace:

$$\mathcal{L}_{\mathrm{global}}(G_i) = \left\| (\mathbf{h}_{G_i}^{\mathrm{logic}} - \boldsymbol{\mu}) - \mathbf{U}^\top \mathbf{U} (\mathbf{h}_{G_i}^{\mathrm{logic}} - \boldsymbol{\mu}) \right\|_2^2,$$

An instance-level anchoring loss pulls each logical embedding toward its paired text embedding:

$$\mathcal{L}_{\mathrm{instance}}(G_i) = 1 - \cos\left(\mathbf{h}_{G_i}^{\mathrm{logic}}, \mathbf{t}_{G_i}\right).$$

The final objective is: $\mathcal{L}_{\mathrm{align}}(G_i) = \mathcal{L}_{\mathrm{global}}(G_i) + \mathcal{L}_{\mathrm{instance}}(G_i)$.

To incorporate logical structure into the LLM, LoGIST treats the aligned logic embedding as an auxiliary conditioning signal. Rather than directly adding it to the token representations, which may disrupt the LLM's original text representations, we use a cross-attention-based logic fusion module [1]. At selected decoder layers, the textual hidden states attend to the aligned logic embedding, and the resulting logic-conditioned update is added back through a residual connection scaled by a learnable gate. This allows the model to selectively use logical information while preserving the original token representations. Specifically, at decoder layer ℓ, let $\mathbf{H}_i^\ell \in \mathbb{R}^{T\times d_t}$ denote the textual hidden states for input i, where T is the sequence length. The textual hidden states are used as queries,

while the aligned logic embedding $\mathbf{h}_{G_i}^{\text{logic}}$ provides the keys and values:

$$\mathbf{Q}_i^\ell = \mathbf{W}_Q^\ell \,\text{LayerNorm}(\mathbf{H}_i^\ell), \qquad \mathbf{K}_i^\ell = \mathbf{W}_K^\ell \,\mathbf{h}_{G_i}^{\text{logic}}, \qquad \mathbf{V}_i^\ell = \mathbf{W}_V^\ell \,\mathbf{h}_{G_i}^{\text{logic}},$$

where $\mathbf{W}_Q^\ell, \mathbf{W}_K^\ell, \mathbf{W}_V^\ell$ are learned projections. The attention output is then projected and added to the original textual states through a gated residual update:

$$\widetilde{\mathbf{H}}_i^\ell = \mathbf{H}_i^\ell + \alpha_\ell \,\mathbf{W}_O^\ell \,\text{Attn}(\mathbf{Q}_i^\ell, \mathbf{K}_i^\ell, \mathbf{V}_i^\ell),$$

where $\mathbf{W}_O^\ell$ is a learned output projection and α_ℓ is a learnable scalar gate for layer ℓ, which controls the strength of the logic-conditioned update.

5 Experiment

In this section, we describe the basic experimental setup, including the dataset and evaluation metrics used. Based on them, we investigate three key research questions: **RQ1**: Does LoGIST produce consistent predictions across logically equivalent inputs? **RQ2**: Can LoGIST improve out-of-distribution generalization in logical reasoning? **RQ3**: Can LoGIST serve as an effective logic encoding strategy for incorporating logical information into LLMs?

5.1 Experimental Setup

Datasets. We evaluate our method in two complementary settings. First, we test structural generalization under logical equivalence using a synthetic reasoning dataset. The training set contains 1,500 instances built from implicational rules. We next generate another 900 examples using the same procedure to form a base evaluation set, and then apply the transformations in Sec. 3.1 to obtain four out-of-distribution reformulations that are logically equivalent to the base examples. Second, we conduct additional training and evaluation on real-world benchmarks, including FOLIO [18] and ProverQA [39], to assess effectiveness beyond our controlled setting. For example, human experts wrote FOLIO. For FOLIO, we follow the standard split. For ProverQA, we use the split that provides paired formal representations, partitioned into 1,000 training and 500 evaluation instances.

Training Details. To evaluate out-of-distribution (OOD) generalization to unseen logical forms, we adopt a strict training protocol. We first train the logic encoder solely on ROBDDs from the synthetic implicational training set, then freeze it and jointly train the LLM and the logic fusion module. The resulting model serves as the starting point. On the synthetic dataset, the LLM is trained only on the implicational training set and evaluated on the base evaluation set and four transformed variants, testing if the learned structure generalizes to unseen logical forms. For FOLIO and ProverQA, we keep the same synthetic-trained logic encoder frozen, train the LLM on each dataset's training set, and evaluate on its corresponding test set. We apply LoGIST to `Llama-3.1-8B-Ins` [49] and `Qwen2.5-3B-Ins` [60] under three training paradigms: SFT with CoT supervision, PPO [45], and GRPO [46]. In each setting, we compare a baseline against its LoGIST-augmented counterpart under identical configurations. Further training details are provided in App. A.3.

5.2 RQ1: LoGIST Improves Consistency Across Logically Equivalent Inputs

We first evaluate whether LoGIST stabilizes LLM predictions across logically equivalent variants using the synthetic dataset. Following the setup described above, we train the LLMs only on the implicational training set, and then compute the logic inconsistency rate by comparing predictions on the base set with those on each transformed set, averaging over three runs per dataset. As shown in Tab. 2, LoGIST nearly halves the average inconsistency rate compared with standard training, reducing it from above 0.3 to roughly 0.15 under both PPO and GRPO, with consistent gains under SFT. The reduction holds across all four transformations,

Table 2: Logic Inconsistency rate across logically equivalent transformations on `Llama-3.1-8B-ins`. A lower score indicates more coherent reasoning.

Methods	Mat.Impl.	Double Neg.	De Morgan	Mixed
SFT	25.7%	18.4%	17.9%	19.0%
SFT(LoGIST)	**20.1**%	**14.2**%	**13.1** %	**15.8**%
PPO	31.5%	29.7%	32.0%	30.6%
PPO(LoGIST)	**15.3**%	**15.9**%	**15.9**%	**13.9**%
GRPO	38.1%	32.5%	34.7%	37.1%
GRPO(LoGIST)	**19.9**%	**20.1**%	**22.4**%	**16.6**%

indicating that it is not driven by any single type of variation. This suggests that LoGIST improves prediction consistency through logical structure rather than surface patterns, enabling generalization to logical forms unseen during training.

5.3 RQ2: LoGIST Improves Generalization in Logical Reasoning

Table 3: Performance on the evaluation set for `Llama-3.1-8B-Ins` and `Qwen2.5-3B-Ins`, with and without LoGIST, across different training strategies. "/" indicates unavailable SFT results on FOLIO due to missing CoT labels. Colored cells indicate better performance. LoGIST consistently improves performance across settings.

LLM	Methods	Synthetic					FOLIO	ProverQA
		Base	Mat.Impl.	Double Neg.	De Morgan	Mixed		
Llama3.1-8B-ins	+ SFT	0.683	0.588	0.625	0.631	0.635	/	0.598
	+ SFT(LoGIST)	0.693	0.604	0.650	0.647	0.651	/	0.638
	+ PPO	0.710	0.614	0.621	0.590	0.618	0.606	0.568
	+ PPO(LoGIST)	0.708	0.655	0.657	0.645	0.642	0.645	0.602
	+ GRPO	0.702	0.608	0.612	0.600	0.627	0.621	0.626
	+ GRPO(LoGIST)	0.730	0.654	0.661	0.647	0.668	0.650	0.652
Qwen2.5-3B-Ins	+ SFT	0.662	0.579	0.610	0.601	0.604	/	0.582
	+ SFT(LoGIST)	0.674	0.607	0.638	0.616	0.630	/	0.624
	+ PPO	0.690	0.631	0.645	0.618	0.636	0.596	0.594
	+ PPO(LoGIST)	0.681	0.661	0.658	0.658	0.657	0.606	0.608
	+ GRPO	0.645	0.570	0.577	0.572	0.590	0.621	0.600
	+ GRPO(LoGIST)	0.655	0.604	0.607	0.598	0.620	0.631	0.616

We next examine whether improved logical invariance leads to stronger generalization. We evaluate accuracy on both synthetic and real-world datasets. As shown in Tab. 3, LoGIST consistently improves performance across nearly all algorithms and datasets. On the synthetic dataset, the gains are largest on OOD logical forms, showing that structure learned from implicational training data transfers to logically equivalent forms unseen during training. We also find that the gains are more pronounced for RL-based methods: for example, on `Llama-3.1-8B-Ins`, LoGIST improves GRPO's performance by 0.046 on Mat.Impl. and 0.041 on Mixed, suggesting that learning logical structure may be particularly beneficial when training relies on terminal rewards rather than step-level CoT supervision. On FOLIO and ProverQA, LoGIST also improves accuracy, indicating that the synthetic-trained logic encoder captures structural information useful for real-world reasoning tasks. Together, these results show that learning from logically equivalent structures improves LLM generalization, and that the consistency gains in Sec. 5.2 translate into meaningful accuracy improvements rather than merely reducing prediction flips. We further show in App. A.5 that LoGIST remains effective even in noisy, practical settings where grounded formal reasoning is unavailable.

5.4 RQ3: LoGIST is an Effective Logic Encoding Strategy

We further examine whether LoGIST's gains stem simply from providing the LLM with additional logical structure information in the form of an ROBDD, rather than from how this information is encoded and integrated. To test this, we introduce a baseline that receives the complete ROBDD as text. Following [13], we linearize the diagram into a node-by-node description and include it in the prompt alongside the context (Fig. 7). This baseline therefore has access to the same ROBDD as LoGIST; the two approaches differ only in how the structure is provided: as prompt text versus as a learned logic embedding fused through cross-attention.

Table 4: Performance comparison across different strategies incorporating ROBDDs information. LoGIST achieves better performance.

Method	Base	M.I.	D.N.	D.M.	Mixed
SFT	0.683	0.588	0.625	0.631	0.635
SFT + Linear ROBDD	0.674	0.594	0.642	0.615	0.639
SFT + LoGIST	**0.693**	**0.604**	**0.650**	**0.647**	**0.651**

As shown in Tab. 4, providing the linearized ROBDD as prompt text yields only marginal and inconsistent changes in accuracy (*e.g.*, $+0.4\%$ on Mixed, -0.9% on Base, and -1.6% on D.M.).

Moreover, it does not reduce the inconsistency and instead increases it from 19.0% to 21.3% in Mixed. We further repeat this comparison under PPO and GRPO, where the performance gap becomes even larger. With only terminal rewards available, the LLM struggles to learn how to effectively utilize the verbalized ROBDD information provided in the prompt. These results demonstrate that LoGIST's performance gains stem from its effective utilization of logical structure, rather than merely from access to additional logical information.

5.5 Analysis: Composition as a Linear Direction

Having shown the effectiveness of our trained logic encoder, we next examine how logical structure is reflected in its representation space. Specifically, we examine how the composition of consecutive reasoning steps is represented in this space by comparing the resulting logic embedding with the sum of their individual embeddings. We ask whether this composition follows a consistent geometric pattern across instances. Motivated by prior work linking linear structure in representation spaces to generalization [59, 51, 34, 19, 26], we test whether the composed embedding can be approximated by this sum plus a shared linear shift in the learned logical embedding space.

To test this hypothesis, we measure whether adding a reasoning step behaves approximately linearly in the learned embedding space. For each instance i with N reasoning steps, we construct three ROBDDs at step n: G_i^n, which contains only the n-th rule; $G_i^{<n}$, which contains the rules preceding step n; and $G_i^{\leq n}$, which contains all rules up to and including step n. Let $\mathbf{h}^{\text{logic}}_{G_i,n}$, $\mathbf{h}^{\text{logic}}_{G_i,<n}$, and $\mathbf{h}^{\text{logic}}_{G_i,\leq n}$ denote the corresponding logic embeddings. We define the compositional residual at step n as: $\boldsymbol{\delta}_i^n = \mathbf{h}^{\text{logic}}_{G_i,\leq n} - \left(\mathbf{h}^{\text{logic}}_{G_i,<n} + \mathbf{h}^{\text{logic}}_{G_i,n}\right)$. This residual captures the part of the composed representation that is not explained by simply adding the representation of the preceding rules and that of the isolated n-th rule. If the same reasoning step follows a shared compositional pattern across instances, then the residuals $\boldsymbol{\delta}_i^n$ should align along a common direction rather than vary independently.

To test whether the residuals indeed concentrate on a shared direction, we adopt the leave-one-out (LOO) alignment score [35]: for each instance, we check whether its residual aligns with the mean direction estimated from all other instances. Specifically, we first whiten each residual to reduce representation anisotropy: $\tilde{\boldsymbol{\delta}}_i^n = \boldsymbol{\Omega}\left(\boldsymbol{\delta}_i^n - \hat{\boldsymbol{\mu}}\right)$, where $\hat{\boldsymbol{\mu}}$ is the mean of the pooled residuals and $\boldsymbol{\Omega}$ is a whitening operator. We then normalize each whitened residual as $\hat{\tilde{\boldsymbol{\delta}}}_i^n = \frac{\tilde{\boldsymbol{\delta}}_i^n}{\|\tilde{\boldsymbol{\delta}}_i^n\|_2}$, and compute the LOO mean direction by averaging over all other instances as $\mathbf{u}_{-i}^n = \frac{\sum_{j\neq i} \hat{\tilde{\boldsymbol{\delta}}}_j^n}{\left\|\sum_{j\neq i} \hat{\tilde{\boldsymbol{\delta}}}_j^n\right\|_2}$. The LOO alignment score is then the inner product $\left\langle \hat{\tilde{\boldsymbol{\delta}}}_i^n, \mathbf{u}_{-i}^n \right\rangle$, which is high when the residual of instance i is well explained by the shared direction induced by the rest.

To verify that this linear compositional structure is specific to our logic encoder rather than a generic property of LLM representations, we compare LOO alignment scores between two representations: the aligned logic embedding $\mathbf{h}^{\text{logic}}_{G_i}$, and the mean-pooled final-layer LLM hidden state of the corresponding natural-language input. Although the two vectors come from different sources, they share the same dimensionality and can thus be directly compared.

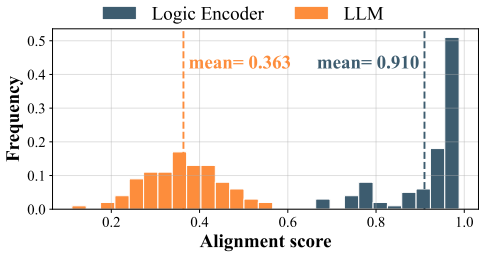


Figure 5: Distributional comparison of the per-instance LOO alignment scores for `Llama-3.1-8B-Ins` and its logic encoder.

As shown in Fig. 5, the logic encoder yields high per-instance LOO alignment scores concentrated near 0.9 (mean $= 0.910$), whereas the LLM text representations score much lower (mean $= 0.363$). This suggests that the shared residual direction is learned by the logic encoder rather than already present in the LLM representation space. This direction is also stable across reasoning depths: the LOO mean directions estimated at different reasoning steps have an average pairwise cosine similarity of 0.90, indicating that the same compositional pattern is reused from one step to the next. Together, these results suggest that the logic encoder captures reasoning-step composition through a shared linear structure, providing evidence for its ability to support generalization across logical forms.

We further include a study of LoGIST's efficiency in App. A.2, where we show that LoGIST introduces only minimal overhead and negligible additional latency. We also demonstrate LoGIST's robustness in App. A.5, showing that it maintains its performance even when ground-truth formal logic representations are unavailable.

6 Related Work

LLMs for Formal Reasoning Formal reasoning systems such as Lean [30], Z3 [10], and Prover9 [27] have long provided verifiable inference with guaranteed soundness. Recent work extends LLMs to formal reasoning along two directions. Data-driven methods improve LLMs' reasoning capability through curated training data [18, 47, 44], or through synthesized logical reasoning data [39]. These approaches optimize task-level accuracy without modeling logical structure, limiting generalization on unseen logical forms. External-tool methods couple LLMs with symbolic solvers to improve reasoning reliability [33, 62, 31], with recent work further enhancing solver use through semantic verification across alternative translations [43] or sub-proposition feedback for iterative proof correction [7]. While effective, these methods rely on external systems for formal reasoning, leaving the LLM's own structure-aware reasoning capability underexplored. In contrast, we show that learning from logical equivalence can improve generalizable logical reasoning in LLMs.

Multimodal LLMs Extending LLMs beyond pure text to additional modalities has long been an active research field. [40, 8, 20] study large-scale vision-language pretraining and cross-modal alignment, while [24, 1] further connect visual features with pretrained language models for multimodal understanding. Beyond visual inputs, recent work has explored modalities related to symbolic information: [57] benchmarks symbolic reasoning in LLMs using multimodal inputs; [22] uses graphs as an additional modality for joint graph-text representation learning; [21] encodes table schemas to enhance tabular reasoning; [17, 11] encode code structure and syntax as a modality to improve LLMs' understanding of code semantics; and [36] exploits code symmetries for code-LLM training. However, to our knowledge, our work is the first to model logical structure to guide logic-invariant reasoning in LLMs and further improve their generalizability.

7 Conclusion

In this work, we identify a key limitation of current LLMs: strong reasoning performance does not necessarily imply a precise understanding of the underlying logical structure. Through systematic evaluation on logically equivalent reformulations, we show that even strong LLMs produce inconsistent predictions and exhibit degraded performance when the same logic is expressed in different forms, limiting their ability to generalize to unseen logical forms. To address this limitation, we propose LoGIST, a logic-structure-guided framework that maps each instance to a decision diagram, encodes it into a structure-aware logic embedding through contrastive learning, and conditions LLM reasoning on this embedding. Across different training paradigms and both synthetic and real-world benchmarks, LoGIST consistently reduces logical inconsistency and improves generalization to unseen logical forms. Further analysis shows that LoGIST encodes logical composition as a linear direction, which helps explain the observed gains in generalization. These results suggest that logical invariance is an effective target for building more robust and generalizable LLM reasoning systems.

Acknowledgement

We thank the anonymous reviewer for their effort in helping us revise our study. This work was supported in part by the National Science Foundation under CAREER Grant No. IIS-2543142 and Grant No. DMS-2412577. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh,

Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[2] László Babai and Eugene M. Luks. Canonical labeling of graphs. In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 171–183. ACM, 1983.

[3] Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on "a is b" fail to learn "b is a". In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[4] Karl S. Brace, Richard L. Rudell, and Randal E. Bryant. Efficient implementation of a bdd package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, DAC '90, page 40–45, New York, NY, USA, 1991. Association for Computing Machinery.

[5] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.

[6] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[7] Chuxue Cao, Mengze Li, Juntao Dai, Jinluan Yang, Zijian Zhao, Shengyu Zhang, Weijie Shi, Chengzhong Liu, Sirui Han, and Yike Guo. Towards advanced mathematical reasoning for llms via first-order logic theorem proving. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 12429–12449. Association for Computational Linguistics, 2025.

[8] Xi Chen, Xiao Wang, Soravit Changpinyo, A. J. Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish V. Thapliyal, James Bradbury, and Weicheng Kuo. Pali: A jointly-scaled multilingual language-image model. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[9] I.M. Copi and C. Cohen. *Introduction to Logic*. Maxwell Macmillan international editions. Macmillan, 1990.

[10] Leonardo De Moura and Nikolaj Bjørner. Z3: an efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, page 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] Yangruibo Ding, Jinjun Peng, Marcus J. Min, Gail E. Kaiser, Junfeng Yang, and Baishakhi Ray. Semcoder: Training code language models with comprehensive semantics reasoning. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[12] Kevin Du, Vésteinn Snæbjarnarson, Niklas Stoehr, Jennifer C. White, Aaron Schein, and Ryan Cotterell. Context versus prior knowledge in language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 13211–13235. Association for Computational Linguistics, 2024.

[13] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Melvin Fitting. *First-Order Logic and Automated Theorem Proving, Second Edition*. Graduate Texts in Computer Science. Springer, 1996.

[15] Melvin Fitting. Modal logics between propositional and first-order. *J. Log. Comput.*, 12(6):1017–1026, 2002.

[16] Michael Genesereth and Eric Kao. *Introduction to Logic*. Morgan & Claypool Publishers, 2012.

[17] Linyuan Gong, Mostafa Elhoushi, and Alvin Cheung. AST-T5: structure-aware pretraining for code generation and understanding. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 15839–15853. PMLR / OpenReview.net, 2024.

[18] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. FOLIO: natural language reasoning with first-order logic. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 22017–22031. Association for Computational Linguistics, 2024.

[19] Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[20] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pages 4904–4916. PMLR, 2021.

[21] Liyao Li, Chao Ye, Wentao Ye, Yifei Sun, Zhe Jiang, Haobo Wang, Jiaming Tian, Yiming Zhang, NINGTAO WANG, Xing Fu, Gang Chen, and Junbo Zhao. Table as a modality for large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[22] Jiacheng Lin, Kun Qian, Haoyu Han, Nurendra Choudhary, Tianxin Wei, Zhongruo Wang, Sahika Genc, E-Wen Huang, Sheng Wang, Karthik Subbian, Danai Koutra, and Jimeng Sun. Gt2vec: Large language models as multi-modal encoders for text and graph-structured data. 2024.

[23] Steven Lindell. A logspace algorithm for tree canonization (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 400–404. ACM, 1992.

[24] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[25] Xuyuan Liu, Shengyu Chen, Xinshuai Dong, Yanchi Liu, Xujiang Zhao, Haoyu Wang, Yujun Yan, Haifeng Chen, and Zhengzhang Chen. Representation interventions enable lifelong knowledge memory control in llms. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pages 5414–5436. Association for Computational Linguistics, 2026.

[26] Xuyuan Liu, Lei Hsiung, Yaoqing Yang, and Yujun Yan. Spectral insights into data-oblivious critical layers in large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pages 4860–4877. Association for Computational Linguistics, 2025.

[27] William McCune. Prover9 and mace4. `https://www.cs.unm.edu/~mccune/prover9/`, 2005.

[28] Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[29] Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. Learning deductive reasoning from synthetic corpus based on formal logic. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 25254–25274. PMLR, 2023.

[30] Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The lean theorem prover (system description). *Automated Deduction – CADE-25*, 9195:378–388, 2015.

[31] Theo X. Olausson, Alex Gu, Ben Lipkin, Cedegao E. Zhang, Armando Solar-Lezama, Joshua B. Tenenbaum, and Roger P. Levy. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[32] OpenAI. Introducing openai o3 and o4-mini. `https://openai.com/index/introducing-o3-and-o4-mini/`, April 2025.

[33] Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore, December 2023. Association for Computational Linguistics.

[34] Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. ICLR: in-context learning of representations. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[35] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 39643–39666. PMLR / OpenReview.net, 2024.

[36] Kexin Pei, Weichen Li, Qirui Jin, Shuyang Liu, Scott Geng, Lorenzo Cavallaro, Junfeng Yang, and Suman Jana. Exploiting code symmetries for learning program semantics. In *Forty-first International Conference on Machine Learning*, 2024.

[37] Gabriel Poesia, Kanishk Gandhi, Eric Zelikman, and Noah D. Goodman. Certified deductive reasoning with language models. *Trans. Mach. Learn. Res.*, 2024, 2024.

[38] Rizky Ramadhana Putra, Raihan Sultan Pasha Basuki, Yutong Cheng, and Peng Gao. NL2Logic: AST-guided translation of natural language into first-order logic with large language models. In Vera Demberg, Kentaro Inui, and Lluís Marquez, editors, *Findings of the Association for Computational Linguistics: EACL 2026*, pages 6035–6051, Rabat, Morocco, March 2026. Association for Computational Linguistics.

[39] Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. Large language models meet symbolic provers for logical reasoning evaluation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[40] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pages 8748–8763. PMLR, 2021.

[41] Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[42] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill Higher Education, 5th edition, 2002.

[43] Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[44] Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. Testing the general deductive reasoning capacity of large language models using OOD examples. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[45] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[46] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.

[47] Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, Findings of ACL, pages 3621–3634. Association for Computational Linguistics, 2021.

[48] Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261, 2025.

[49] Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

[50] Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025.

[51] Matthew Trager, Pramuditha Perera, Luca Zancato, Alessandro Achille, Parminder Bhatia, and Stefano Soatto. Linear spaces of meanings: Compositional structures in vision-language models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 15349–15358. IEEE, 2023.

[52] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[53] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, pages 9929–9939. PMLR, 2020.

[54] Xinyi Wang, Antonis Antoniades, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. Generalization v.s. memorization: Tracing language models' capabilities back to pretraining data. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[55] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[56] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[57] Jundong Xu, Hao Fei, Yuhui Zhang, Liangming Pan, Qijun Huang, Qian Liu, Preslav Nakov, Min-Yen Kan, William Yang Wang, Mong-Li Lee, and Wynne Hsu. Muslr: Multimodal symbolic logical reasoning. *CoRR*, abs/2509.25851, 2025.

[58] Yujun Yan, Kevin Swersky, Danai Koutra, Parthasarathy Ranganathan, and Milad Hashemi. Neural execution engines: Learning to execute subroutines. *Advances in Neural Information Processing Systems*, 33:17298–17308, 2020.

[59] Yujun Yan, Kevin Swersky, Danai Koutra, Parthasarathy Ranganathan, and Milad Hashemi. Neural execution engines: Learning to execute subroutines. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[60] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024.

[61] Yiran Yang, Haifeng Sun, Jingyu Wang, Qi Qi, Zirui Zhuang, Huazheng Wang, Pengfei Ren, Jing Wang, and Jianxin Liao. Unveiling internal reasoning modes in llms: A deep dive into latent reasoning vs. factual shortcuts with attribute rate ratio. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 2186–2206. Association for Computational Linguistics, 2025.

[62] Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[63] Yu Yuan, Lili Zhao, Kai Zhang, Guangting Zheng, and Qi Liu. Do llms overcome shortcut learning? an evaluation of shortcut challenges in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 12188–12200. Association for Computational Linguistics, 2024.

[64] Xinyue Zeng, Jiawei Zhang, Yujun Yan, and Dawei Zhou. Sage: Mitigating long-horizon reasoning biases via topological guidance, 2026.

[65] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[66] Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. SELF-DISCOVER: large language models self-compose reasoning structures. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[67] Yue Zhou, Yada Zhu, Diego Antognini, Yoon Kim, and Yang Zhang. Paraphrase and solve: Exploring and exploiting the impact of surface form on mathematical reasoning in large language models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2793–2804. Association for Computational Linguistics, 2024.

A Appendix

A.1 Logical Encoder Algorithm

Algorithm 1 Algorithm for ROBDD Construction

Require: Formal premises $\Gamma^{\mathrm{FL}} = \mathcal{F} \cup \mathcal{R}$, entities $\mathcal{E}$, and ground query $\mathcal{Q}^{\mathrm{FL}}$
Ensure: ROBDD G with root ρ, terminals τ_0, τ_1, and low/high edge sets E_0/E_1
// Stage 1: Grounding
$\Gamma_{\mathrm{g}} \leftarrow \mathrm{Ground}_{\mathcal{E}}(\Gamma^{\mathrm{FL}})$ ▷ ground all rules over the instance's entities
$F \leftarrow \bigwedge_{g \in \Gamma_{\mathrm{g}}} g$ ▷ each ground atom is a Boolean variable
// Stage 2: Variable Ordering
$L \leftarrow$ BACKWARDDFS$(\mathrm{atoms}(\mathcal{Q}^{\mathrm{FL}}), \Gamma_{\mathrm{g}})$ ▷ recursively follow fact/rule dependencies; visit each atom once
$(x_1, \ldots, x_m) \leftarrow L \parallel \mathrm{Sort}(\mathrm{atoms}(\Gamma_{\mathrm{g}}) \setminus \mathrm{set}(L))$
Use shared dependencies and fixed traversal/sorting orders across equivalent variants.
// Stage 3: Shannon Expansion and Reduction
Initialize G with τ_0, τ_1 and $E_0 = E_1 = \varnothing$
function BUILD(F, i)
  if $F \equiv c \in \{0, 1\}$ **then return** τ_c
  $u_b \leftarrow$ BUILD$(F|_{x_i=b}, i+1)$ for $b \in \{0, 1\}$ ▷ Shannon expansion
  if $u_0 = u_1$ **then return** u_0 ▷ remove redundant nodes
  return existing node (x_i, u_0, u_1) in G, or create one with low/high successors u_0/u_1 ▷ merge isomorphic subgraphs
end function
$\rho \leftarrow$ BUILD$(F, 1)$
return G rooted at ρ

Algorithm 2 Algorithm for Logical Encoder

Require: ROBDD graph G with terminal nodes $\{\tau_0, \tau_1\} \subseteq V$; high/low branch set E_0/E_1; target LLM embedding layer $\mathrm{Emb}(\cdot)$; edge-type-specific projections $\{\mathbf{W}_g^k, \mathbf{W}_m^k\}_{k \in \{0,1\}}$, where $\mathbf{W}_g^k$ produces the gate and $\mathbf{W}_m^k$ produces the message for edge type k
Ensure: Graph-level representation $\mathbf{h}_G \in \mathbb{R}^{d_g}$.
// Stage 1: Initialization
for each $v \in V$ **do**
  $\mathbf{h}_v \leftarrow \mathrm{Emb}(v)$
end for
// Stage 2: Topological Ordering
Compute in-degree for each node and sort V into topological layers $L_1, L_2, \ldots, L_T$
// Stage 3: Layer-wise Message Passing
for $t = 1$ to T **do**
  for each $v \in L_t$ with in-degree$(v) = 0$ in remaining graph **do**
    for $k \in \{0, 1\}$ **do**
      $m_k(v) \leftarrow \sum_{(u,v) \in E_k} \sigma(\mathbf{W}_g^k \mathbf{h}_u) \odot (\mathbf{W}_m^k \mathbf{h}_u)$
    end for
    $\mathbf{h}_v \leftarrow \mathrm{GRU}(m_0(v) + m_1(v),\ \mathbf{h}_v)$
  end for
  Remove processed nodes and update in-degrees
end for
// Stage 4: Graph-level Readout
$\mathbf{h}_G \leftarrow \mathrm{MLP}([\mathbf{h}_{\tau_0} \| \mathbf{h}_{\tau_1}])$
return $\mathbf{h}_G$

A.2 Study of Efficiency

We next quantify the parameter overhead introduced by LoGIST on top of the base LLM. Let L_{LLM} denote the number of decoder layers in the base LLM and d_t its hidden dimension. For logic fusion, we insert a gated cross-attention module every g decoder layers, resulting in $\lceil L_{\mathrm{LLM}}/g \rceil$ cross-attention modules in total. For the logic encoder, let L_g denote the number of message-passing layers and d_g the graph hidden dimension, which is used for both node states $\mathbf{h}_v$ and the graph-level readout $\mathbf{h}_{G_i}$.

LoGIST introduces learnable parameters through three components: (i) the cross-attention modules, which incorporate the logic representation into the LLM hidden states; (ii) the graph encoder backbone, which maps the input logic graph into a structure-aware representation; and (iii) the graph-to-text aligner, implemented as a two-layer MLP, which projects graph representations into the LLM hidden space.

Each cross-attention module operates in dimension d_t and contributes $4\,d_t^2 + 6\,d_t$ parameters. Summing over the $\lceil L_{\mathrm{LLM}}/g \rceil$ inserted modules gives

$$P_{\mathrm{cross}} = \left\lceil \frac{L_{\mathrm{LLM}}}{g} \right\rceil \left(4\,d_t^2 + 6\,d_t\right).$$

Each graph encoder layer contains two edge-type-specific linear transformations and one GRU-based update, contributing $8\,d_g^2 + 8\,d_g$ parameters. Summing over the L_g layers gives

$$P_{\mathrm{enc}} = L_g\big(8\,d_g^2 + 8\,d_g\big).$$

The aligner adds

$$P_{\mathrm{align}} = d_g\,d_t + d_t^2 + 2\,d_t.$$

The total parameter overhead is therefore

$$\begin{aligned} P_{\mathrm{LoGIST}} &= P_{\mathrm{cross}} + P_{\mathrm{enc}} + P_{\mathrm{align}} \\ &= 2\left\lceil \frac{L_{\mathrm{LLM}}}{g} \right\rceil d_t(2\,d_t + 3) + 8\,L_g\,d_g(d_g + 1) + d_t(d_g + d_t + 2). \end{aligned}$$

Concretely, in our setup, `Llama-3.1-8B-Ins` introduces 312M additional parameters and `Qwen2.5-3B-Ins` introduces 101M, both amounting to only about 3% of the full model size. Most of the added parameters are concentrated in the cross-attention modules rather than the graph encoder, so the design remains parameter-efficient. Moreover, since the training pipeline freezes the graph encoder by default, the number of trainable additional parameters is strictly smaller than the total overhead reported above.

We further compare the inference speed of each base model with its LoGIST-enhanced variant. Under identical generation configurations and on the same device, we report the mean generation throughput, measured in generated tokens per second (token/s), over 500 examples in Table 5.

Table 5: Inference-speed comparison between the original models and their LoGIST-enhanced variants. LoGIST introduces minimal additional overhead compared with the base model.

Model	`Llama-3.1-8B-Ins`	`Qwen2.5-3B-Ins`
w/o LoGIST	58.20 (token/s)	56.76 (token/s)
w/ LoGIST	56.99 (token/s)	55.70 (token/s)

This result is expected: the logical embedding remains static during generation and is computed only once at the prefilling stage, so the main runtime overhead comes from the cross-attention modules, which add less than 2% extra cost. LoGIST is therefore an efficient framework for incorporating logical structure into the LLM to guide structure-aware reasoning.

A.3 Training Setup

LoGIST is algorithm-agnostic and can be plugged into existing LLM training pipelines. We apply it to `Llama-3.1-8B-Ins` [49] and `Qwen2.5-3B-Ins` [60] under three training paradigms: SFT with CoT supervision, PPO [45], and GRPO [46]. For SFT, we use CoT responses as training labels; for PPO and GRPO, we use a binary reward based on exact matching of the final answer. Across all three paradigms, the logic encoder is kept frozen after its contrastive pretraining on the synthetic implicational dataset, and only the LLM and the logic fusion module are updated.

To isolate the contribution of LoGIST from dataset-specific tuning, we use the same hyperparameters across all datasets within each paradigm rather than tuning them per dataset. The full configurations for `Llama-3.1-8B-Ins` are reported in Table 6.

Table 6: Training configurations for the three training paradigms (SFT, PPO, GRPO) on `Llama-3.1-8B-Ins`.

(a) SFT

Configuration	Value
Learning Rate(LR)	1e-5
Batch Size	128
Epochs	3
Gate Value (α_ℓ)	5e-2
Layer Interval g	8
Graph Dimension	512
Cross-Atten LR	5e-6

(b) PPO

Configuration	Value
Actor Learning Rate	1e-6
Critic Learning Rate	5e-6
Batch Size	144
KL Coefficient	0.02
Epochs	3
Gate Value (α_ℓ)	5e-2
Layer Interval g	8
Graph Dimension	512
Cross-Atten LR	5e-6

(c) GRPO

Configuration	Value
Actor Learning Rate	1e-6
Batch Size	144
Rollout Size	5
KL Coefficient	0.02
Epochs	3
Gate Value (α_ℓ)	5e-2
Layer Interval g	8
Graph Dimension	512
Cross-Atten LR	5e-6

For the data used in SFT, we show an example below in Fig 6. In this setup, the Reasoning Response explicitly states the dependency structure required for reasoning.

A.4 Rules of Replacement

We list below all rules of replacement used to generate logically equivalent variants.

Table 7: Rules of replacement used in our transformations and their corresponding tautologies.

Rule of Replacement	Tautology
MATERIAL IMPLICATION	$(p \to q) \Leftrightarrow (\neg p \lor q)$
DOUBLE NEGATION INTRO	$p \Leftrightarrow \neg\neg p$
DE MORGAN'S LAW 1	$\neg(p \land q) \Leftrightarrow (\neg p) \lor (\neg q)$
DE MORGAN'S LAW 2	$\neg(p \lor q) \Leftrightarrow (\neg p) \land (\neg q)$

A.5 Robustness of LoGIST under Practical Setting

In practice, a formal representation is not always available and can be parsed from the natural-language input, so ROBDDs can be imperfect due to parsing errors such as mismatched predicate names, missing or extra literals, or incorrect connectives. We therefore directly study this realistic setting.

To examine whether LoGIST depends on gold annotations, we ran a new experiment in which the formal representation at evaluation time is instead extracted automatically from the natural-language text, following NL2Logic [38]. Notably, the extraction is performed by the vanilla Llama-3.1-8B-Ins model, the same base model used for training, rather than any specialized logic model. The parsed FOL is genuinely noisy: on the Mixed data, only 15.3% of the parsed representations exactly match the gold FOL (mostly due to predicate naming differing from the gold annotations), while 73.6% are structurally correct. The model is trained following the pipeline as in the paper and evaluated with ROBDDs built from this noisy parsed FOL. The results are shown in Tab. 8.

SFT Data Example.

Instruction: You will be given a logical reasoning question with context information. Analyze the given information carefully and determine the correct answer from the provided options.
Context:
Brennan is creative.
Brennan does not have obstacles.
Brennan meets adversity.
If Brennan is creative and faces challenges, then he produces inspirational work.
Anyone who has obstacles or meets adversity faces challenges.

Question:
Based on the above information, is the following statement true, false, or uncertain? *Brennan does not produce inspirational work.*
Answer: B

Reasoning Response (SFT Label) Brennan does not have obstacles (fact 1) but he does meet adversity (fact 2). According to the rule, anyone who has obstacles or meets adversity faces challenges. Therefore, since Brennan meets adversity, he faces challenges.\n\nAdditionally, it is stated that Brennan is creative (fact 1) and that he faces challenges (fact 2). The rule indicates that if Brennan is creative and faces challenges, then he produces inspirational work. However, the conclusion drawn is that Brennan does not produce inspirational work.\n\nThis leads to a contradiction, as it is false to assert that Brennan does not produce inspirational work given the conditions established. \n\n Therefore, the correct option is: B.

Figure 6: An example of SFT training data, where the reasoning response is used as the training label to explicitly teach the LLM the dependency structure required for reasoning.

Table 8: Performance comparison across different settings.

Method	Base	M.I.	D.N.	D.M.	Mixed
GRPO + LoGIST (Gold FOL)	0.730	0.654	0.661	0.647	0.668
GRPO + LoGIST (Constructed FOL)	0.722	0.649	0.650	0.645	0.656
Vanilla GRPO	0.702	0.608	0.612	0.600	0.627

Our results show that even under this realistic noise level, LoGIST still yields stable improvements: when the ROBDD is imperfect, the resulting performance change is negligible and the model stays close to its gold-FOL input performance. This indicates that LoGIST remains effective with realistic, automatically parsed representations and does not rely on noise-free annotations. We will add this experiment and discussion to the revised manuscript.

A.6 Study: Contrastive Training Enables Generalizable Logical Structure Learning

Table 9: Performance comparison with and w/o contrastive learning for `Qwen2.5-3B-Ins` trained with GRPO.

Methods	Base	Mat.Impl.	Double Neg.	De Morgan	Mixed
w/ Contrastive	0.655	0.604	0.607	0.598	0.620
w/o Contrastive	0.651	0.571	0.584	0.578	0.595
GRPO Baseline	0.645	0.570	0.577	0.572	0.590

We finally examine how contrastive training enables the logic encoder to capture structural information that supports generalizable reasoning. To isolate its effect, we compare three variants that differ only in encoder initialization: **(i)** LoGIST with the contrastively trained encoder from Sec. 4.2, **(ii)** LoGIST with a randomly initialized encoder (no contrastive training), and **(iii)** the GRPO baseline. As shown in Tab. 9, LoGIST without contrastive training yields only marginal gains over the GRPO baseline, whereas the contrastively trained encoder produces consistent and substantial improvements across all settings. This comparison indicates that the structural information needed for generalizable reasoning

is successfully acquired during contrastive training, and the resulting logic embeddings encode reusable structural features that the LLM can leverage to generalize across logically equivalent inputs.

A.7 Prompt Template

This section presents the basic prompt used in the paper.

Prompt Template for Logical Reasoning

Instruction: You will be given a logical reasoning question with context information. Analyze the given information carefully and determine the correct answer from the provided options.

Context:
<CONTEXT>

Question:
Based on the above information, is the following statement true, false, or uncertain?

<STATEMENT>

Options:
A) True
B) False
C) Uncertain

Let's think step by step:

Linearized ROBDD Prompt Template for Logical Reasoning

Instruction:
You will be given contextual information, a linearized reasoning graph, and a query statement. Carefully analyze the provided information and determine whether the statement is **true**, **false**, or **uncertain**.

Context:
<CONTEXT>

Reasoning Graph (Linearized ROBDD):
```
Root: N0
N0: if contribute_to_film_preservation(Aurora) then N2 else FALSE
N2: if appreciate_film_history(Aurora) then N3 else FALSE
N3: if collect_videodisks(Aurora) then FALSE else N4
N4: if study_cinema(Aurora) then TRUE else FALSE
```

Question:
Based on the context and reasoning graph above, is the following statement **true**, **false**, or **uncertain**?
<STATEMENT>

Options:
A) True
B) False
C) Uncertain

Let's think step by step:

Figure 7: Linearized ROBDD prompt template for logical reasoning.

We also provide the prompt used to guide LLMs toward recognizing equivalent logical structures, as described in Sec. 3.2.

Prompt Template for Learning Logically Equivalent Structures

Instruction. You will be given a logical reasoning question with contextual information. Analyze the information carefully and determine the correct answer from the provided options.

Hints.
Commutative Laws

- Use this law when the same terms appear in a different order within a conjunction or disjunction.
- $A \wedge B \equiv B \wedge A$.
- $A \vee B \Leftrightarrow B \vee A$.

Material Implication Law

- Use this law when rewriting an if–then statement as an equivalent disjunction.
- $A \rightarrow B \Leftrightarrow \neg A \vee B$.

Double Negation Elimination

- Use this law when a proposition contains two consecutive negations.
- $\neg\neg A \Leftrightarrow A$.

De Morgan's Laws

- Use this law when a negation applies to an entire conjunction or disjunction and needs to be pushed inward.
- $\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$.
- $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$.

Key Insight. Apply these equivalence laws only when they help simplify the context and support the derivation of the correct answer.

Context.
<CONTEXT>

Question.
<QUESTION>

Options:
A) True
B) False
C) Uncertain

Answer. Let's think step by step.

Figure 8: Prompt template used for prompting equivalent logical learning.

A.8 Data Example of Equivalent Logic

In this part, we present an example of our Equivalent Logic transformation. We start by constructing the initial data using implicational rule templates, and then apply a rule of replacement to the rule set in the premise to obtain a logically equivalent variant. In this process, the transformed version shares the same fact set as the initial one, while the transformed rule set is presented to the LLM.

Initial Facts and Rules

- Aurora does not collect videodisks. $\neg\text{collect_videodisks}(\text{Aurora})$
- Aurora studies cinema. $\text{study_cinema}(\text{Aurora})$
- If Aurora collects videodisks or appreciates film history, then she contributes to film preservation. $(\text{collect_videodisks}(\text{Aurora}) \lor \text{appreciate_film_history}(\text{Aurora})) \rightarrow \text{contribute_to_film_preservation}(\text{Aurora})$
- Anyone who watches classics or studies cinema can appreciate film history. $\forall x\big((\text{watch_classics}(x) \lor \text{study_cinema}(x)) \rightarrow \text{appreciate_film_history}(x)\big)$

Material Implication

- It is not the case that Aurora collects videodisks or appreciates film history, or she contributes to film preservation. $\neg(\text{collect_videodisks}(\text{Aurora}) \lor \text{appreciate_film_history}(\text{Aurora})) \lor \text{contribute_to_film_preservation}(\text{Aurora})$
- For all people, it is not the case that they watch classics or study cinema, or they appreciate film history.$\forall x\big(\neg(\text{watch_classics}(x) \lor \text{study_cinema}(x)) \lor \text{appreciate_film_history}(x)\big)$

De Morgan's Law

- If Aurora does not both fail to collect videodisks and fail to appreciate film history, then she contributes to film preservation. $\neg\big(\neg\text{collect_videodisks}(\text{Aurora}) \land \neg\text{appreciate_film_history}(\text{Aurora})\big) \rightarrow \text{contribute_to_film_preservation}(\text{Aurora})$
- For all people, if they do not both fail to watch classics and fail to study cinema, then they can appreciate film history. $\forall x\big(\neg(\neg\text{watch_classics}(x) \land \neg\text{study_cinema}(x)) \rightarrow \text{appreciate_film_history}(x)\big)$

Double Negation Intro

- If it is not the case that Aurora fails to either collect videodisks or appreciate film history, then she contributes to film preservation. $\neg\neg\big(\text{collect_videodisks}(\text{Aurora}) \lor \text{appreciate_film_history}(\text{Aurora})\big) \rightarrow \text{contribute_to_film_preservation}(\text{Aurora})$
- For all people, if it is not the case that they fail to either watch classics or study cinema, then they can appreciate film history. $\forall x\big(\neg\neg(\text{watch_classics}(x) \lor \text{study_cinema}(x)) \rightarrow \text{appreciate_film_history}(x)\big)$

Mixed Transformation

- If Aurora collects videodisks or appreciates film history, then she contributes to film preservation. $(\text{collect_videodisks}(\text{Aurora}) \lor \text{appreciate_film_history}(\text{Aurora})) \rightarrow \text{contribute_to_film_preservation}(\text{Aurora})$
- For all people, if it is not the case that they fail to either watch classics or study cinema, then they can appreciate film history. $\forall x\big(\neg\neg(\text{watch_classics}(x) \lor \text{study_cinema}(x)) \rightarrow \text{appreciate_film_history}(x)\big)$

Figure 9: An example of logically equivalent transformations applied to a fact set and its associated rule sets. The fact set is kept fixed, while the rule set is rewritten under different equivalence transformations. The De Morgan case requires an implicit Double Negation Introduction step before De Morgan's Law can be applied, since De Morgan's Law strictly rewrites a negated conjunction or disjunction.

A.9 Limitation and Potential Impact

In this work, we unveil a key limitation of current LLMs in capturing logical structure when performing reasoning tasks, and present a framework showing that explicitly encoding logical structure can improve model reasoning generalization. This finding has potential impact for applications that require faithful symbolic reasoning, such as formal verification, theorem proving assistance, program analysis, and high-stakes decision support where consistency under equivalent logical formulations is important.

Despite these promising results, our study has limitations. First, we focus on formal reasoning in this work; extending LoGIST to domains such as code reasoning, where structural factors go beyond Boolean equivalence, e.g., data-flow dependencies and variable scoping, remains future work. Second, our current implementation prioritizes feasibility and empirical validation; integrating LoGIST into mainstream inference engines such as vLLM and SGLang for higher-throughput deployment remains future engineering work.