LATENT STRUCTURE AS A COMPLEMENTARY PREDICTOR FOR LANGUAGE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Most neural reasoning architectures produce task predictions through a single classifier, even when their internal representations encode relational information. We investigate whether latent structure induced without structural annotation can support a complementary predictor alongside direct neural classification. We develop a dual-path architecture that constructs latent structure over token-ID groups. Multiple context-dependent prototypes summarize the occurrences of each token, prototype compatibility defines interactions between groups, and gated mixture-of-experts message passing produces group representations for structured prediction. In parallel, a neural branch predicts directly from contextual token states. An input-dependent gate combines the two outputs using the margin between the two largest latent-state logits as a fusion signal. We evaluate the architecture from random initialization against similarly sized decoder-style models under matched model-selection and evaluation protocols on FOLIO, JustLogic, ProofWriter, ANLI-R1, SNLI, and MNLI. It obtains the highest observed mean accuracy among the evaluated models on all six datasets. On FOLIO, a ten-seed comparison with an independently trained neural-only variant retaining the same shared encoder and memory path yields a mean gain of 1.58 percentage points, with a 95% confidence interval of $[0.08, 3.07]$. Single-seed branch-level diagnostics further show that fusion can correct errors made by the neural branch. These results provide evidence that task-induced latent structure can function as a distinct and complementary source of predictions in language reasoning.

1 INTRODUCTION

Neural reasoning models typically make downstream predictions from contextual token representations (Devlin et al., 2019). Latent-structure models have further shown that task supervision can induce internal structure without direct structural annotation and use it to construct task representations (Corro & Titov, 2019; Choi et al., 2018). In representative latent-structure approaches, however, the induced structure typically serves as an intermediate representation or computation mechanism for a downstream predictor rather than producing task-level predictions through a separate classification head (Williams et al., 2018a). This leaves an open question: can task-supervised latent structure form a separate prediction branch that complements conventional neural classification?

Existing work has introduced structure into language reasoning in several ways. One line of research maps text to logical forms or executable programs (Liang et al., 2017; Pasupat & Liang, 2016; Dong & Lapata, 2016), proof objects (Lin et al., 2022), knowledge-base queries (Agarwal et al., 2024), AMR graphs (Flanigan et al., 2014), database queries (Suhr et al., 2020), proof-assistant specifications (Wu et al., 2022), predefined concepts (Koh et al., 2020), or explanation trees (Dalvi et al., 2021). Another line learns latent trees or grammars (Drozdov et al., 2019; Kim et al., 2019; Yang et al., 2021; Choi et al., 2018), latent relations or reasoning structures (Liu & Lapata, 2018; Nan et al., 2020; Labutov et al., 2018), or latent compositional representations (Yin et al., 2018; Kim et al., 2017) from weak or task supervision. Probing studies instead examine the syntactic and semantic information encoded in contextual representations (Tenney et al., 2019; Conia & Navigli, 2022). Related approaches introduce structured computation through abductive learning (Dai et al., 2019), verifier-guided reasoning (Yang et al., 2022; Li et al., 2023a), neural modules (Andreas et al., 2016), differentiable logic (Badreddine et al., 2022), or latent grounding (Liu et al., 2021). Together, these methods demonstrate the value of structure in language reasoning, but they either rely on

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

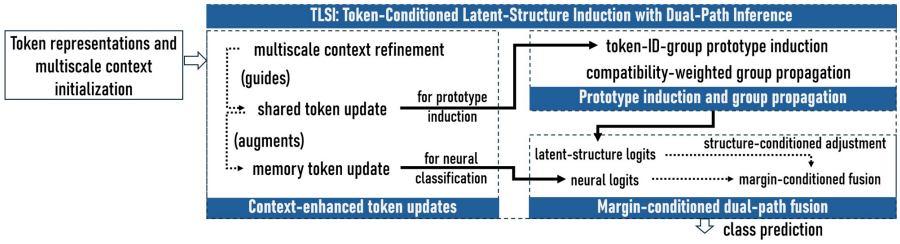

Figure 1: Overview of TLSI. An adopted shared multiscale encoder produces token states used by the neural and latent-structure prediction branches introduced in this paper.

predefined representations, external execution or verification mechanisms, or do not construct task-induced latent structure as a separate prediction branch.

To study this possibility, we introduce Token-Conditioned Latent-Structure Induction with Dual-Path Inference (TLSI). TLSI induces multiple context-dependent occurrence prototypes within each token-ID group, constructs group-level interactions from prototype compatibility, and propagates one state per token-ID group through mixture-of-experts message passing. The latent-structure branch pools and classifies the propagated group states, while the neural branch produces class logits directly from the contextual token states. An input-dependent fusion gate then combines the two logits, with its latent-structure branch weight modulated by the mean top-two margin of the latent-state logits across token-ID groups. This design directly tests whether task-supervised latent structure can provide information complementary to conventional neural classification. *We compare similarly sized, randomly initialized decoder-style architectures under matched protocols, reflecting TLSI's causal token-update path and final-token neural readout. This controlled setting isolates architectural differences from pretraining effects, and our conclusions therefore do not extend to pretrained systems.*

Contributions. **First**, we introduce TLSI, which uses task-supervised latent structure to construct a separate prediction branch alongside direct neural classification. **Second**, across six logical-inference and natural-language-inference datasets, TLSI achieves the highest observed mean accuracy among the evaluated randomly initialized models under matched evaluation protocols. **Third**, a ten-seed, same-encoder neural-only control on FOLIO provides controlled evidence for the incremental contribution of the latent-structure pathway, while single-seed component ablations and branch-level diagnostics further examine its behavior.

2 METHOD

Overview. Let $\mathcal{S} = \{0, 1, 2, 3\}$ be the context scales; let $x = (x_1, \ldots, x_L)$ have valid positions $\mathcal{I}(x)$, observed token IDs $\mathcal{V}(x) = \{x_i : i \in \mathcal{I}(x)\}$, and occurrence sets $G_v(x) = \{i \in \mathcal{I}(x) : x_i = v\}$. TLSI builds on the complex-vector token representation and token–context attention framework of Zhang & Sheng (2026) and adopts a multiscale token–context encoder from concurrent work (Anonymous, 2026). This encoder produces token states and the refined multiscale context states used by the two prediction branches. The latent-structure branch induces multiple context-dependent prototypes within each token-ID group and propagates one group representation per token ID. In parallel, the neural classification branch augments the contextual token representations with token-ID-indexed memory. Separate classification heads produce structured and neural logits, which are combined by an input-dependent fusion gate. All induced prototypes, affinities, group states, and latent states are continuous variables learned from task supervision. TLSI does not recover logical forms, proof objects, discrete logical relations, or externally grounded symbols.

2.1 CONTEXT-ENHANCED TOKEN UPDATES

Multiscale feature construction from input sequences Because token semantics is context-dependent and can be disambiguated from its surrounding context (Li et al., 2010), we first construct multiscale context for each token occurrence in the input sequences.

For each two-segment input, let c and q denote two input segments. For $u \in \{c, q\}$, let T_u denote its padded sequence length and D denote the embedding dimension. Let $E^u \in \mathbb{R}^{T_u \times D}$ and $m^u \in \{0,1\}^{T_u}$ denote the padded embedding sequence and active-position mask, respectively. Let $\mathcal{K}_u$ denote the sequence-specific set of convolution kernel sizes. For each position j in input sequence u, the multiscale token–context encoder computes the projected token representation $x_j^u = m_j^u(W_u e_j^u + b_u)$. The resulting sequence $X^u = (x_1^u, \ldots, x_{T_u}^u)$ is then processed by a bank of convolutional operators to produce multiscale convolutional features: $F_{u,k} = \mathrm{ReLU}(\mathrm{Conv}_{u,k}(X^u))$ for $k \in \mathcal{K}_u$.

These features form four multiscale components: a width-1 local component $C_{u,0} = F_{u,1}$, two learned mixtures $C_{u,s} = \sum_{k \in \mathcal{K}_{u,s}} \gamma_{u,s,k} F_{u,k}$ for $s \in \{1,2\}$, with $\mathcal{K}_{u,1} = \mathcal{K}_u \setminus \{1\}$ and $\mathcal{K}_{u,2} = \mathcal{K}_u$, and a global component obtained by projecting the sequence-wise mean of the input features. The mixture coefficients $\gamma_{u,s,k}$ are normalized Gaussian weights whose centers depend on learned fractions of the running mean sequence length. This running mean is updated during training and fixed during evaluation.

Following Zhang & Sheng (2026), we map each component $C_{u,s}$ to a complex-vector token representation $Z_{u,s}$. Its real part encodes the initial token representation, and its imaginary part is retained as a scale-specific context representation: $H_j^{u,(0)} = \sum_{s=0}^{3} \omega_{u,s} \,\mathrm{Re}(Z_{u,s,j}) + P_j^u$ and $C_j^{u,(s,0)} = \mathrm{Im}(Z_{u,s,j}) + P_j^u$, where $\omega_{u,s}$ are learned component-fusion weights and P^u is a sinusoidal positional encoding. Representations at padded positions are set to zero.

We then concatenate these sequences in their original order to form $H^{(0)} = [H^{c,(0)}; H^{q,(0)}]$ and $C^{(s,0)} = [C^{c,(s,0)}; C^{q,(s,0)}]$, yielding $C^{(0)} \in \mathbb{R}^{L \times 4 \times D}$. Following the dependency-locality motivation of Futrell et al. (2015), we use denser coverage of short receptive fields while retaining several larger kernels for longer-range interactions. Because the context sequence has a larger maximum length than the hypothesis sequence, its candidate kernel bank spans a wider range. Specifically, we use $\mathcal{K}_c = \{1, 3, 7, 13, 21, 33\}$ and $\mathcal{K}_q = \{1, 3, 5, 7, 9, 13\}$. Thus, $\mathcal{K}_u$ specifies the available receptive fields, whereas $\gamma_{u,s,k}$ determines the relative contribution of each receptive field to component s.

Token-ID-conditioned multiscale context refinement. Token occurrences sharing the same token ID usually preserve a common lexical identity, and their contexts tend to induce more similar token-specific semantics than those of different tokens (Ethayarajh, 2019). We therefore refine the contexts within each token-ID group using a self-attention-based Transformer.

For each scale $s \in \mathcal{S}$, let $C^{(s,0)} \in \mathbb{R}^{L \times D}$ denote the initial context sequence. A scale-specific Transformer refines this sequence using the additive attention mask M: $C^{(s,\mathrm{grp})} = \mathrm{Transformer}_s(C^{(s,0)}, M)$, where, for $i \in \mathcal{I}(x)$, $M_{ij} = 0$ if $j \in \mathcal{I}(x)$ and $x_i = x_j$, and $M_{ij} = -\infty$ otherwise. Invalid positions are zeroed. TLSI then uses the residual-gating scalar $\gamma_{\mathrm{cc}}^{(s)}$ to form the intermediate context state $c_i^{(s,\mathrm{mid})} = c_i^{(s,0)} + \tanh(\gamma_{\mathrm{cc}}^{(s)})(c_i^{(s,\mathrm{grp})} - c_i^{(s,0)})$, yielding the context tensor $C^{\mathrm{mid}} = \{c_i^{(s,\mathrm{mid})}\}_{i \in \mathcal{I}(x),\, s \in \mathcal{S}}$.

Context refinement via slot-based feature extraction. After token-ID-group-level refinement, the contexts remain occurrence-level features and may still contain noise and redundant variations. Inspired by slot-based representation learning (Locatello et al., 2020), we compress them into a small set of scale-specific contextual prototypes. These prototypes summarize recurring contextual patterns at each scale and serve as compact semantic references.

Let L_{slot} be the number of slot-attention layers, and initialize the occurrence-specific context state as $c_{i,0}^{(s)} = c_i^{(s,\mathrm{mid})}$. Let K_{ctx} denote the number of slots per scale and d_a the attention-head dimension. In each of the $L_{\mathrm{slot}} = 12$ layers, slots attend to valid positions at the same scale with $\eta_{k,i,\ell}^{(s)} = \mathrm{softmax}_i(\langle q_{k,\ell}^{(s)}, k_{i,\ell}^{(s)} \rangle / \sqrt{d_a})$ and are updated by attention and feed-forward residual blocks. Each position then attends to the updated slots through $b_{i,k,\ell}^{(s)} = \mathrm{softmax}_k(\langle c_{i,\ell-1}^{(s)}, \tilde{s}_{k,\ell}^{(s)} \rangle / \sqrt{D})$ and $p_{i,\ell}^{(s)} = \sum_{k=1}^{K_{\mathrm{ctx}}} b_{i,k,\ell}^{(s)} \tilde{s}_{k,\ell}^{(s)}$, followed by the residual-gated update $c_{i,\ell}^{(s)} = c_{i,\ell-1}^{(s)} + \tanh(\gamma_s) p_{i,\ell}^{(s)}$. The output is $c_{i,\mathrm{slot}}^{(s)} = c_{i,L_{\mathrm{slot}}}^{(s)}$ and $C^{\mathrm{slot}} = \{c_{i,\mathrm{slot}}^{(s)}\}_{i \in \mathcal{I}(x),\, s \in \mathcal{S}}$.

Relational refinement of contextual slots. Next, we refine the contextual slots by modeling interactions across slots and scales, and map the relation-enhanced slot information back to each context position.

Index each slot–scale node by $a = (s, k)$, and let $u_a^{(0)}$ be its projected slot state. The initial edge state is $e_{a,a'}^{(0)} = \mathrm{LN}_e\left(f_e\left([u_a^{(0)}; u_{a'}^{(0)}; u_a^{(0)} \odot u_{a'}^{(0)}; u_a^{(0)} - u_{a'}^{(0)}]\right)\right)$. We use the edge states to bias node self-attention and update the node and edge states with separate feed-forward residual blocks. Let $\hat{s}_k^{(s)}$ denote the relationally refined representation obtained from the resulting slot state $\tilde{s}_{k,L_{\mathrm{slot}}}^{(s)}$. The recomputed assignment weights and the resulting context representation are $b_{i,k}^{(s),\mathrm{rel}} = \mathrm{softmax}_k(\langle c_{i,\mathrm{slot}}^{(s)}, \hat{s}_k^{(s)}\rangle/\sqrt{D})$ and $p_{i,\mathrm{rel}}^{(s)} = \sum_{k=1}^{K_{\mathrm{ctx}}} b_{i,k}^{(s),\mathrm{rel}} \hat{s}_k^{(s)}$, respectively. We define slot-attention and slot-diversity regularizers in the training objective.

Context-guided token update for latent-structure induction. Prior work Zhang & Sheng (2026) suggests that computing attention weights from token-context interactions can reduce reliance on surface-form matching. Motivated by this observation, we use the refined multiscale context as scale-specific guidance for token updates.

Let $H^{(\ell-1)} = (h_{1,\ell-1}, \ldots, h_{L,\ell-1}) \in \mathbb{R}^{L\times D}$ denote the token-state sequence entering layer ℓ, where $h_{i,\ell-1} \in \mathbb{R}^D$ is the token state at position i. The initial sequence is $H^{(0)}$, so $h_{i,0} = h_i^{(0)}$. Following the adopted multiscale conditioning mechanism (Anonymous, 2026), let $f_{j,\ell}^{(s)}$ denote the scale-specific conditioning feature supplied to layer ℓ at position j. The query and value are shared across scales, while the key is scale-specific: $q_{i,\ell} = W_{Q,\ell}\,\mathrm{LN}_{\mathrm{tok},\ell}(h_{i,\ell-1})$, $k_{j,\ell}^{(s)} = W_{K,\ell}^{(s)}\left(c_{j,\mathrm{slot}}^{(s)} + \xi_{\ell,s} f_{j,\ell}^{(s)}\right)$, and $v_{i,\ell} = W_{V,\ell}\,\mathrm{LN}_{\mathrm{tok},\ell}(h_{i,\ell-1})$, respectively, where $W_{Q,\ell}$, $W_{K,\ell}^{(s)}$, and $W_{V,\ell}$ are the learned query, key, and value projection matrices, and $\xi_{\ell,s}$ is a learned parameter. The masked attention weights and context-guided update at scale s are $a_{ij,\ell}^{(s)} = \mathrm{softmax}_j\left(\langle q_{i,\ell}, k_{j,\ell}^{(s)}\rangle/\sqrt{d_a} + \mu_{ij}\right)$ and $u_{i,\ell}^{(s)} = \sum_{j=1}^{L} a_{ij,\ell}^{(s)} v_{j,\ell}$, where $\mu_{ij} = 0$ for a valid position $j \le i$ and $\mu_{ij} = -\infty$ otherwise. The scale-specific updates are concatenated and passed through the output projection, residual connection, and feed-forward block.

The output token sequence $H^{\mathrm{shared}} = (h_1^{\mathrm{shared}}, \ldots, h_L^{\mathrm{shared}})$ is the shared representation from which both prediction branches are constructed. The latent-structure branch uses it to construct token-ID-group prototypes, while the neural branch augments it with token-ID-indexed context memory. The context representations $c_{i,\mathrm{slot}}^{(s)}$ and $p_{i,\mathrm{rel}}^{(s)}$ are retained for the memory module.

Token-ID-indexed context memory for neural classification head. To save possible reusable features observed across previous input texts for contextual semantics, we augment the multiscale contexts with a small bank of retrieved token-ID-indexed context memory.

For each token ID v, the online context memory has capacity P_{mem} and stores prototypes $M_{v,p} \in \mathbb{R}^{S\times D}$, where $S = |\mathcal{S}|$ is the number of context scales and D is the hidden dimension of each scale-specific context representation. At position i, the learned residual-gating scalar γ_{def} gives $c_{i,\mathrm{def}}^{(s)} = c_{i,\mathrm{slot}}^{(s)} + \tanh(\gamma_{\mathrm{def}})(p_{i,\mathrm{rel}}^{(s)} - c_{i,\mathrm{slot}}^{(s)})$ and $c_{i,\mathrm{def}} = \mathrm{Stack}_{s\in\mathcal{S}}(c_{i,\mathrm{def}}^{(s)})$. Then $Y_i^{\mathrm{mem}} = \mathrm{Stack}(c_{i,\mathrm{def}}, M_{x_i,1}, \ldots, M_{x_i,P_{\mathrm{mem}}}) \in \mathbb{R}^{(1+P_{\mathrm{mem}})\times S\times D}$, where the first axis indexes the current context and memory candidates, and the second axis indexes the context scales. We then use one message-passing step to update scales within each candidate, and another message-passing across token, candidate, and scale nodes, with cross-token communication restricted to matching token IDs. Token-conditioned attention over the resulting tensor $\widetilde{Y}_i^{\mathrm{mem}}$ yields the memory update δ_i. With training step t starting at 1, $w_{\mathrm{ctx}}(t) = \mathrm{clip}(t/200, 0, 1)$ during training and $w_{\mathrm{ctx}} = 1$ during evaluation, giving $h_i^{\mathrm{neu}} = \mathrm{LN}\left(h_i^{\mathrm{shared}} + w_{\mathrm{ctx}}(t)\tanh(\gamma_{\mathrm{ctx}})\delta_i\right)$

The context memory is initialized for each seed, indexed only by token IDs observed in the training split, read before each mini-batch update, updated after both prediction branches, and frozen during evaluation. Let $b_{v,p} \in \{0, 1\}$ indicate whether memory slot p for token ID v is valid. For occurrences with token ID v, the current-context slices $\widetilde{c}_{i,\mathrm{def}} = \widetilde{Y}_{i,0,:,:}^{\mathrm{mem}}$ first fill

invalid memory slots, and the remaining tensors form $\mathcal{C}_v$. For similarity computation, each $c \in \mathcal{C}_v$ and memory prototype $M_{v,p}$ are vectorized, and c is assigned to its most similar valid slot: $p^\star(c) = \arg\max_{\substack{1\le p\le P_{\mathrm{mem}} \\ b_{v,p}=1}} \cos(\mathrm{vec}(c), \mathrm{vec}(M_{v,p}))$. For each nonempty assignment set $\mathcal{C}_{v,p} = \{c \in \mathcal{C}_v : p^\star(c) = p\}$, we compute $\bar{c}_{v,p} = |\mathcal{C}_{v,p}|^{-1} \sum_{c\in\mathcal{C}_{v,p}} c$ and update the corresponding prototype by $M_{v,p} \leftarrow (1-\lambda_{\mathrm{mem}})M_{v,p} + \lambda_{\mathrm{mem}}\bar{c}_{v,p}$, where $\lambda_{\mathrm{mem}} = 0.05$. Only valid slots associated with token IDs represented in the memory index participate in retrieval.

2.2 Latent-Structure Induction and Propagation

Occurrence prototypes within token-ID groups. Prior work on word-in-context evaluation treats the meaning of a shared target word as context-dependent rather than determined by word identity alone (Pilehvar & Camacho-Collados, 2019). Motivated by this distinction, TLSI represents each token-ID group using multiple contextualized occurrences rather than collapsing the group into a single embedding.

Using the shared pre-memory sequence H^{shared}, TLSI constructs $R_v = \min(K_{\mathrm{occ}}, |G_v(x)|)$ soft prototypes for each token ID v, where K_{occ} is the per-group prototype cap. A learned scorer initializes $a_{v,1,i}$. The selector distribution is $\mathbf{g}_{v,r} = \mathrm{softmax}_{i\in G_v(x)}(a_{v,r,i}/\tau_{\mathrm{sel}})$, with component $g_{v,r,i}$. The selector logits are then updated as $a_{v,r+1,i} = a_{v,r,i} + \log(1 - g_{v,r,i} + \epsilon)$. Here $\tau_{\mathrm{sel}} > 0$ is the selector temperature and $\epsilon > 0$ is a numerical constant. This recurrence downweights previously emphasized positions, encouraging successive selectors to attend to different occurrences. The occurrence prototype is $t_{v,r} = \sum_{i\in G_v(x)} g_{v,r,i} h_i^{\mathrm{shared}}$. The group-level prototype summary is $\bar{t}_v = R_v^{-1} \sum_{r=1}^{R_v} t_{v,r}$.

Compatibility-based group propagation. Next, TLSI aggregates the occurrence prototypes into one state per token-ID group and refines the resulting group states through compatibility-weighted message passing. Accordingly, K_{occ} bounds the number of occurrence prototypes used to construct each group summary and the compatibility matrix, while TLSI maintains one propagation node for each observed token ID. The initial group state is $h_v^{(0)} = \phi_{\mathrm{init}}(\bar{t}_v)$, where ϕ_{init} is a learned multilayer perceptron. For token-ID groups $v, v' \in \mathcal{V}(x)$, TLSI computes the maximum cosine similarity over valid occurrence-prototype pairs: $\chi_{v,v'} = \max_{\substack{1\le r\le R_v \\ 1\le r'\le R_{v'}}} \cos(t_{v,r}, t_{v',r'})$. The learned compatibility threshold and temperature are $\tau_{\mathrm{cmp}} = \tanh(\theta_\tau)$ and $T_{\mathrm{cmp}} = \mathrm{softplus}(\theta_T) + \epsilon$, respectively. We transform the similarity matrix into $\widetilde{A}_{v,v'} = \sigma\left(\frac{\chi_{v,v'} - \tau_{\mathrm{cmp}}}{T_{\mathrm{cmp}}}\right)$, explicitly enforce symmetry as $A = \frac{\widetilde{A} + \widetilde{A}^\top}{2}$, and exclude self-edges by setting $A_{v,v} = 0$. Let K_{exp} denote the number of propagation experts and let $\mathcal{E} = \{1, \ldots, K_{\mathrm{exp}}\}$. We use $K_{\mathrm{exp}} = 5$, and all experts operate on the same compatibility matrix A and the same group-level prototype summary $\bar{t}_v$. Each expert has distinct parameters shared across propagation steps.

At propagation step j, each expert index $\kappa \in \mathcal{E}$ combines the current group state with the group-level prototype summary to update an expert-specific intermediate state: $x_v^{(\kappa,j)} = \mathrm{LN}_\kappa\left(h_v^{(j)} + W_{\mathrm{ctx}}^{(\kappa)} \bar{t}_v\right)$. The projected value state is $y_v^{(\kappa,j)} = W_{\mathrm{val}}^{(\kappa)} x_v^{(\kappa,j)}$. Using the shared compatibility matrix A, the expert computes $r_v^{(\kappa,j)} = \sum_{v'\in\mathcal{V}(x)} A_{v,v'} y_{v'}^{(\kappa,j)}$. The corresponding expert update is $e_v^{(\kappa,j)} = f_\kappa\left([h_v^{(j)}; \bar{t}_v; r_v^{(\kappa,j)}]\right)$. We define the composite gating map as $\phi_{\mathrm{gate}}(h,t) = f_{\mathrm{gate}}(\mathrm{LN}_{\mathrm{gate}}([h;t]))$. The mixture-gate temperature is $T_{\mathrm{gate}} = \mathrm{softplus}(\theta_{\mathrm{gate}}) + \epsilon$, and we represent the input-dependent expert weights as $\lambda_v^{(j)} = \mathrm{softmax}\left(\phi_{\mathrm{gate}}\left(h_v^{(j)}, \bar{t}_v\right)/T_{\mathrm{gate}}\right) \in \mathbb{R}^{K_{\mathrm{exp}}}$. The combined expert update is then $e_v^{(j)} = \sum_{\kappa\in\mathcal{E}} \lambda_{v,\kappa}^{(j)} e_v^{(\kappa,j)}$. We compute the next group state as $\widetilde{h}_v^{(j)} = \mathrm{LN}_1(h_v^{(j)} + \mathrm{Dropout}(e_v^{(j)}))$ and $h_v^{(j+1)} = \mathrm{LN}_2(\widetilde{h}_v^{(j)} + \mathrm{FFN}(\widetilde{h}_v^{(j)}))$. TLSI applies $J = 3$ propagation steps, then the final structured group state is $h_v^{\mathrm{str}} = h_v^{(J)}$. Each final group state produces scores over K_{lat} latent states. These states are learned end-to-end from task supervision and are used only as internal structural features for logit adjustment and branch fusion. We set $K_{\mathrm{lat}} = 4$ in all experiments. The corresponding latent-state logits are $\ell_v^{\mathrm{state}} = f_{\mathrm{state}}(h_v^{\mathrm{str}}) \in \mathbb{R}^{K_{\mathrm{lat}}}$.

Table 1: Controlled random-initialization comparison of approximately one-billion-parameter architectures.

Logical inference datasets

Model	Dim	FOLIO		JustLogic-mixed		ProofWriter-mixed	
		Acc	Paired Δ [95% CI]	Acc	Paired Δ [95% CI]	Acc	Paired Δ [95% CI]
TLSI (0.93B)	768	**47.54**	—	**75.57**	—	**57.35**	—
Llama 3-style (1.23B)	2048	42.41	5.12 [0.45, 9.80]	51.95	23.62 [22.19, 25.05]	49.19	8.16 [5.63, 10.69]
Pythia-style (1.08B)	2048	39.36	8.18 [4.12, 12.23]	*57.28*	18.29 [16.15, 20.43]	*49.51*	7.84 [5.31, 10.36]
Phi 1.5-style (1.3B)	2048	*45.52*	2.02 [-1.75, 5.79]‡	56.26	19.31 [16.84, 21.78]	49.28	8.08 [5.81, 10.34]
Gemma3-style (1B)	1152	45.13	2.41 [0.36, 4.47]†	53.53	22.04 [20.25, 23.83]	49.32	8.03 [5.77, 10.29]

Natural language inference datasets

Model	Dim	ANLI-R1		SNLI		MNLI-mm	
		Acc	Paired Δ [95% CI]	Acc	Paired Δ [95% CI]	Acc	Paired Δ [95% CI]
TLSI (0.93B)	768	**39.54**	—	**65.45**	—	**55.18**	—
Llama 3-style (1.23B)	2048	36.52	3.02 [2.09, 3.95]	59.56	5.89 [5.01, 6.77]	48.95	6.23 [5.46, 6.99]
Pythia-style (1.08B)	2048	35.69	3.85 [2.13, 5.57]	*63.16*	2.29 [0.74, 3.85]	*51.83*	3.34 [2.26, 4.42]
Phi 1.5-style (1.3B)	2048	*36.76*	2.78 [1.78, 3.78]	62.85	2.60 [1.95, 3.26]	51.73	3.45 [2.71, 4.19]
Gemma3-style (1B)	1152	35.65	3.89 [2.30, 5.48]	59.94	5.51 [4.36, 6.66]	49.95	5.23 [4.41, 6.05]

†: 90% CI. ‡: Phi 1.5-style shows greater variation across seeds, with an accuracy of 45.52% ± 4.33%, compared with 47.54% ± 2.08% for TLSI. This larger run-to-run variation helps explain why the paired 95% confidence interval still includes zero, even though TLSI has a higher mean accuracy on FOLIO.

Affinity construction and state smoothing. TLSI converts occurrence-prototype similarities into a soft affinity matrix and uses it to smooth the latent-state distributions supplied to the structure-conditioned logit adjustment.

Let $T = |\mathcal{V}(x)|$. For token-ID groups $v, v' \in \mathcal{V}(x)$, the raw prototype affinity is defined over occurrence-prototype pairs: $s^{\text{aff}}_{v,v'} = \max_{\substack{1\le r\le R_v \\ 1\le r'\le R_{v'}}} \cos\left(t_{v,r}, t_{v',r'}\right)$, where the maximum is taken only over valid occurrence prototypes. We set $\tau_{\text{aff}} = 0.86$ and $T_{\text{aff}} = 0.05$ as fixed hyperparameters. The initial affinity is $\widetilde{E}^{(0)}_{v,v'} = \sigma\left(\frac{s^{\text{aff}}_{v,v'} - \tau_{\text{aff}}}{T_{\text{aff}}}\right)$. TLSI next symmetrizes this matrix and sets its diagonal to one: $E^{(0)}_{v,v'} = \frac{\widetilde{E}^{(0)}_{v,v'} + \widetilde{E}^{(0)}_{v',v}}{2}$ for $v \neq v'$, and $E^{(0)}_{v,v} = 1$.

TLSI then applies two closure-style affinity updates. For $n \in \{0, 1\}$, define $P^{(n)}_{v,v'} = T^{-1} \sum_{w\in\mathcal{V}(x)} E^{(n)}_{v,w} E^{(n)}_{w,v'}$ and $\widetilde{E}^{(n+1)}_{v,v'} = 1 - \left(1 - E^{(n)}_{v,v'}\right)\left(1 - P^{(n)}_{v,v'}\right)$. The updated matrix is symmetrized, clipped to $[0, 1]$ and assigned a unit diagonal: $E^{(n+1)}_{v,v'} = \text{clip}\left(\frac{\widetilde{E}^{(n+1)}_{v,v'} + \widetilde{E}^{(n+1)}_{v',v}}{2}, 0, 1\right)$ for $v \neq v'$, and $E^{(n+1)}_{v,v} = 1$. The final affinity matrix is then $E = E^{(2)}$. The unsmoothed latent-state distribution is $p^{\text{raw}}_v = \text{softmax}\left(\ell^{\text{state}}_v / 0.50\right)$. Because $E_{v,v} = 1$, every row of E has strictly positive mass. Its row-normalized form is $\widehat{E}_{v,v'} = \frac{E_{v,v'}}{\sum_{\tilde{v}\in\mathcal{V}(x)} E_{v,\tilde{v}}}$, and the affinity-smoothed latent-state distribution is $p^{\text{state}}_v = \sum_{v'\in\mathcal{V}(x)} \widehat{E}_{v,v'} p^{\text{raw}}_{v'}$.

The affinity matrix E has a unit diagonal and undergoes two closure-style updates. In contrast, the propagation matrix A uses learned threshold and temperature parameters, has a zero diagonal, and is used directly for message passing without any closure-style transformation.

The neural branch uses the last non-padding token of the query or hypothesis sequence as its pooled representation. If $i^\star$ denotes this position, the neural class logits are $z^{\text{neu}} = f_{\text{neu}}(h^{\text{neu}}_{i^\star})$. The structured branch mean-pools the propagated group states and produces core class logits $z^{\text{str}}_{\text{core}} = f_{\text{str}}\left(|\mathcal{V}(x)|^{-1} \sum_{v\in\mathcal{V}(x)} h^{\text{str}}_v\right)$.

Table 2: Test-set accuracy by observed depth.

(a) JustLogic-bucketed ($n_{\text{seed}} = 5$)

Depth	TLSI	Phi 1.5-style	Llama 3-style	Pythia-style	Gemma 3-style
1–3	**59.11**	45.02	36.83	40.51	37.40
4–5	**64.43**	40.95	31.33	37.62	37.90
6–7	**68.10**	37.24	35.62	34.29	34.10

(b) ProofWriter D5 exact-depth accuracy ($n_{\text{seed}} = 2$)

Model	All	0	1	2	3	4	5
TLSI	**73.65**	**84.69**	**56.69**	**66.05**	**72.90**	**72.85**	**73.21**
Phi-1.5-style	52.16	51.85	53.46	52.20	50.62	49.51	50.34
Llama 3-style	54.10	54.28	53.07	53.87	54.49	61.12	61.36
Pythia-style	52.35	50.25	51.03	56.96	60.92	57.79	58.18
Gemma 3-style	56.33	59.68	53.40	53.78	50.31	54.10	50.61

Structure-conditioned logit adjustment. The structured prediction is derived from the induced latent structures. TLSI therefore summarizes their induced structural properties and maps these statistics to a bounded residual that adjusts the structured logits before fusion with the neural logits.

Let $T = |\mathcal{V}(x)|$ and $J = 3$. For group v, let $\ell^{\text{state}}_{v,(1)}$ and $\ell^{\text{state}}_{v,(2)}$ be the largest and second-largest entries of ℓ^{state}_{v}. The mean top-two latent-state logit margin is $m_{\text{logit}} = T^{-1}\sum_{v\in\mathcal{V}(x)}\left(\ell^{\text{state}}_{v,(1)} - \ell^{\text{state}}_{v,(2)}\right)$. The off-diagonal affinity density is $d_{\text{aff}} = 0$ when $T = 1$ and $d_{\text{aff}} = [T(T-1)]^{-1}\sum_{v\neq v'} E_{v,v'}$ when $T > 1$. The differentiable isolated-node proxy is represented as $s_{\text{iso}} = T^{-1}\sum_v \sigma\left((1.5 - \sum_{v'\neq v} E_{v,v'})/0.5\right)$.

For each latent-state dimension $d \in \{2,3,4\}$, define $\bar{p}_d = T^{-1}\sum_{v\in\mathcal{V}(x)} p^{\text{state}}_{v,d}$. The first component is omitted because the four probabilities sum to one. Enumerate the propagation experts by $\kappa \in \mathcal{E}$. Their mean usage is $\bar{\lambda}_\kappa = (TJ)^{-1}\sum_{v\in\mathcal{V}(x)}\sum_{j=0}^{J-1}\lambda^{(j)}_{v,\kappa}$ The structural feature vector is $\Omega_{\text{str}} = [d_{\text{aff}}, s_{\text{iso}}, m_{\text{logit}}, \bar{p}_2, \bar{p}_3, \bar{p}_4, \bar{\lambda}_1, \ldots, \bar{\lambda}_{K_{\text{exp}}}] \in \mathbb{R}^{K_{\text{exp}}+6}$. For any vector $\boldsymbol{\zeta}$, define $\text{clip}_2(\boldsymbol{\zeta}) = \boldsymbol{\zeta}/\max(1, \|\boldsymbol{\zeta}\|_2)$. A multilayer perceptron with residual scale s_{bias} maps the structural feature vector to a bounded class-logit residual: $\delta_{\text{str}} = \text{clip}_2\left(s_{\text{bias}} f_{\text{bias}}(\Omega_{\text{str}})\right) \in \mathbb{R}^{C_{\text{cls}}}$. We use a fixed empirical scaling coefficient of 0.30 for the bounded residual. The adjusted latent-structure class logits are $z^{\text{str}} = z^{\text{str}}_{\text{core}} + 0.30\,\delta_{\text{str}}$.

Top-two logit-margin gate. The mean top-two margin m_{logit}, computed from the unsmoothed latent-state logits, captures logit separation but is uncalibrated and depends on logit scale. We use the margin in two places. In the structural residual, it is combined with the other structural features to adjust the class logits; in the fusion gate, it determines the weight assigned to the structured branch. Let t denote the optimization-step index. During training, the structured-branch warm-up is $w_{\text{str}}(t) = \text{clip}(t/200, 0, 1)$, and $w_{\text{str}} = 1$ during evaluation. The margin response is $r_{\text{margin}} = \sigma\left((m_{\text{logit}} - 0.20)/0.60\right)$. The fusion coefficient is $\alpha = \sigma(\beta) r_{\text{margin}} w_{\text{str}}$, where β is the fusion-scale parameter, and the final class logits are then $z = (1-\alpha)z^{\text{neu}} + \alpha z^{\text{str}}$.

Training objective. For task label ι, let n_ι be its training count and $N = \sum_{c=1}^{C_{\text{cls}}} n_c$; the class weight is $w_\iota = N/(C_{\text{cls}} n_\iota)$. The fused and structured prediction losses are $\mathcal{L}_{\text{main}} = \text{CE}_w(z, y)$ and $\mathcal{L}_{\text{str}} = \text{CE}_w(z^{\text{str}}, y)$, respectively. We additionally regularize slot attention, slot diversity, assignment consistency, assignment concentration, and assignment balance. The complete definitions of these auxiliary losses and the schedule of $\lambda_{\text{ent}}(e)$ are provided in the supplemental material. The full objective is $\mathcal{L} = \mathcal{L}_{\text{main}} + 0.5\mathcal{L}_{\text{str}} + \lambda_{\text{ent}}(e)\mathcal{L}_{\text{slot-attn}} + 0.05\mathcal{L}_{\text{slot-div}} + \mathcal{L}_{\text{group}} + 0.002\mathcal{L}_{\text{sharp}} + 0.05\mathcal{L}_{\text{balance}}$.

3 Results

Evaluation protocol. We compare TLSI with similarly sized, randomly initialized Llama 3-style (Grattafiori et al., 2024), Pythia-style (Biderman et al., 2023), Phi 1.5-style (Li et al., 2023b), and

Table 3: Component ablations on FOLIO.

Variant	Acc.	Drop
Full model	**49.75**	—
w/o context-guided token update	41.87	7.88
w/o token-ID-indexed context memory	47.78	1.97
w/o context refinement	48.28	1.47
w/o compatibility-based group propagation	43.35	6.40
w/o token-ID-group prototype induction	45.81	3.94
w/o structure-conditioned logit adjustment	46.80	2.95
w/o latent-state-margin fusion gate	47.29	2.46

Table 4: FOLIO accuracy by latent-state-margin quintile.

Bucket	n	Neural	Structured	Fused
Lowest	41	43.90	34.15	**48.78**
Low-mid	40	55.00	52.50	**60.00**
Mid	41	43.90	34.15	**46.34**
High-mid	40	40.00	40.00	**42.50**
Highest	41	41.46	**58.54**	51.22

Gemma 3-style (Team et al., 2024) models on FOLIO (Han et al., 2024), JustLogic (Chen et al., 2025), ProofWriter (Tafjord et al., 2021), ANLI-R1 (Nie et al., 2020), SNLI (Bowman et al., 2015), and MNLI (Williams et al., 2018b). All models use shared data partitions and matched model-selection and evaluation procedures. Training uses a GB200/GH200 platform with 128GB memory and 8 CPU cores.

For each architecture and dataset, we search learning rates $\{3 \times 10^{-5}, 5 \times 10^{-5}, 8 \times 10^{-5}, 1 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$ and dropout rates $\{0.01, 0.05, 0.10\}$ with the same five selection seeds. For each pair and seed, we save the checkpoint with highest validation accuracy and select the pair by the mean of these five maxima before reporting-set evaluation; test sets are never used for model selection.

FOLIO, JustLogic-mixed, ANLI-R1, SNLI, and MNLI use ten runs in total: five selection seeds and five evaluation seeds under the selected configuration. ProofWriter-mixed reports only the five selection seeds. JustLogic-bucketed and ProofWriter-depth reuse the mixed-setting learning rate and dropout but are trained independently. JustLogic-bucketed reports five seeds, whereas ProofWriter-depth reports two seeds because each run on the 20% training subset requires more than 100 hours of wall-clock training time under our hardware configuration; these results should therefore be interpreted as descriptive estimates.

We report pretrained 3B–12B LLM-plus-solver systems on FOLIO as reference comparisons, because approximately 1B backbones did not reliably generate executable inputs for solvers, and solver-based ProofWriter protocols do not match our depth setting.

Dataset partitions. Dataset-specific splits and subsampling protocols are summarized in supplemental material. Within each setting, all models share the same partitions, sampled indices, and sampling seeds, and test sets do not affect model selection or early stopping.

Overall inference performance. Table 1 shows that TLSI achieves the highest observed mean accuracy among the compared approximately one-billion-parameter architectures on all six datasets. All paired mean differences favor TLSI, and their 95% confidence intervals exclude zero except for the FOLIO comparisons with Phi 1.5-style and Gemma 3-style.

Reference comparison with external solver pipelines. Table 5 reports our three-run reimplementations of FoVer (Pei et al., 2025), LINC (Olausson et al., 2023), and Logic-LM (Pan et al., 2023) on the FOLIO reporting examples. These systems generate solver-compatible formalizations with pretrained language models and therefore use a different paradigm from TLSI.

Protocol-specific depth analysis. As shown in Table 2, JustLogic-bucketed uses a separate split and the predefined ranges 1–3, 4–5, and 6–7. ProofWriter uses the D5 split and reports exact required depths 0–5 within D5. TLSI achieves the highest mean accuracy in every JustLogic depth range, with margins over the strongest baseline increasing from 14.09 to 30.86 percentage points across the three ranges. Under the reduced-data ProofWriter protocol, TLSI also obtains the highest aggregate accuracy and the highest accuracy at every exact depth. The margins are substantial for the aggregate result and most depth strata, although the depth-1 margin is smaller.

Table 5: Three-run FOLIO for reimplemented LLM-plus-solver pipelines (end-to-end and executable-only accuracy).

Backbone	FoVer	LINC	Logic-LM
Llama3.2-3B	33.00 / 34.01	41.71 / 41.71	42.36 / 53.16
Llama3.2-8B	35.47 / 36.73	60.43 / 60.43	39.57 / 47.83
Gemma3-4B	33.50 / 34.00	37.60 / 37.60	44.99 / 54.82
Gemma3-12B	33.00 / 35.45	63.55 / 63.55	47.95 / 78.38

Table 6: Contribution of the structured branch on FOLIO.

Metric	Value
Neural-branch accuracy	44.83%
Structured-branch accuracy	43.84%
Fused accuracy	49.75%
Gain over neural branch	4.93 pp
Changed neural predictions	14 / 203
Beneficial changes	10 / 203
Harmful changes	0 / 203
Wrong-to-wrong changes	4 / 203

3.1 Ablation study and diagnostic analysis

Neural-only control. To isolate the contribution of the latent-structure pathway from that of the shared encoder and context-memory architecture, we train a neural-only TLSI variant that retains the same shared encoder and neural memory path but removes the latent-structure prediction pathway and dual-path fusion. On FOLIO, across ten matched seeds, the full model achieves 47.54% accuracy, compared with 45.96% for the neural-only variant. The paired mean difference is 1.58 points (95% t-CI $[0.08, 3.07]$; two-sided paired t-test, $p = 0.041$). This controlled comparison provides multi-seed evidence for an incremental contribution of the latent-structure pathway on FOLIO.

Component ablation. All evaluated component removals lower FOLIO accuracy in the reported run (Table 3). Each variant is retrained from random initialization under the full-model protocol, with only losses specific to the removed component omitted. Because these ablations use one seed, they are interpreted as component-level sensitivity diagnostics rather than estimates of stable effect size.

Latent-structure statistics. In one trained FOLIO model, dual-path fusion raises accuracy from 44.83% to 49.75% (Table 6). Among the 14 neural-branch predictions changed by fusion, 10 changes correct neural-branch errors, four remain incorrect, and none introduce an error. This within-model diagnostic complements, but is distinct from, the independently trained ten-seed neural-only control above. Additional descriptive statistics of the induced latent structures are reported in Appendix C.

Association between state margin and branch accuracy. To examine whether the margin tracks structured-branch accuracy, we divide the examples into five approximately equal-sized buckets according to their mean top-two latent-logit margin. Table 4 shows that structured accuracy peaks in the highest-margin quintile but does not increase monotonically across quintiles. The margin therefore serves as an uncalibrated fusion feature rather than an estimate of structured-branch correctness.

Tokenizer replacement. We replace the default Llama 3 tokenizer with the Pythia tokenizer to assess whether TLSI depends on a particular tokenization scheme. With the learning rate and dropout fixed to those selected for the Llama 3 tokenizer, the two tokenizers achieve similar five-seed mean accuracies: 47.88% for Llama 3 and 47.59% for Pythia. This small difference suggests that the observed performance is not strongly tied to the default tokenizer.

4 Limitations

TLSI's latent structures are not grounded symbols, and token-ID grouping does not guarantee semantic identity across repeated subwords. The fusion gate uses an uncalibrated, logit-scale-dependent margin. GPU constraints limit the ten-seed neural-only control to FOLIO, so its evidence is dataset-specific; component ablations and branch diagnostics use one seed. Depth experiments do not test unseen-depth extrapolation, and architecture comparisons are restricted to randomly initialized, approximately one-billion-parameter decoder models without compute-normalized matching.

AI USE STATEMENT

Generative AI tools were used to aid and polish the writing of this paper, to support the retrieval and discovery of related work, to assist in checking and revising portions of the mathematical formulations and derivations, and to support the verification of portions of the experimental code. All AI-assisted content was independently verified by the authors against the stated assumptions and derivations, and all references identified with AI assistance were checked against the original publications. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The experimental setup, model-selection protocol, and evaluation procedure are described in Section 3. Dataset partitions and subsampling procedures are provided in Appendix B, while the auxiliary training objectives and additional latent-structure diagnostics are documented in Appendices A and C, respectively. The supplementary material contains the source code and configurations used to reproduce the reported experiments.

REFERENCES

Prerna Agarwal, Nishant Kumar, and Srikanta Bedathur. Symkgqa: Few-shot knowledge graph question answering via symbolic program generation and execution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10119–10140, 2024.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39–48, June 2016. doi: 10.1109/CVPR.2016.12.

Anonymous. Title omitted for double-blind review, 2026. Concurrent submission.

Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2021.103649. URL https://www.sciencedirect.com/science/article/pii/S0004370221002009.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pp. 2397–2430. PMLR, 2023.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 632–642, 2015.

Michael K Chen, Xikun Zhang, and Dacheng Tao. Justlogic: A comprehensive benchmark for evaluating deductive reasoning in large language models. *arXiv preprint arXiv:2501.14851*, 2025.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. Learning to compose task-specific tree structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Simone Conia and Roberto Navigli. Probing for predicate argument structures in pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4622–4632, 2022.

Caio Corro and Ivan Titov. Learning latent trees with stochastic perturbations and differentiable dynamic programming. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5508–5521, 2019.

Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. Bridging machine learning and logical reasoning by abductive learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. Explaining answers with entailment trees. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7358–7370, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Li Dong and Mirella Lapata. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33–43, 2016.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1129–1141, 2019.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 55–65, 2019.

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1426–1436, 2014.

Richard Futrell, Kyle Mahowald, and Edward Gibson. Large-scale evidence of dependency length minimization in 37 languages. *Proceedings of the National Academy of Sciences*, 112(33):10336–10341, 2015.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, et al. Folio: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22017–22031, 2024.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=HkE0Nvqlg`.

Yoon Kim, Chris Dyer, and Alexander M Rush. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2369–2385, 2019.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pp. 5338–5348. PMLR, 2020.

Igor Labutov, Bishan Yang, and Tom Mitchell. Learning to learn semantic parsers from natural language supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1676–1690, 2018.

Linlin Li, Benjamin Roth, and Caroline Sporleder. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1138–1147, 2010.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 5315–5333, 2023a.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023b.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23–33, 2017.

Zi Lin, Jeremiah Zhe Liu, and Jingbo Shang. Towards collaborative neural-symbolic graph semantic parsing via uncertainty. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 4160–4173, 2022.

Qian Liu, Dejian Yang, Jiahui Zhang, Jiaqi Guo, Bin Zhou, and Jian-Guang Lou. Awakening latent grounding from pretrained language models for semantic parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 1174–1189, 2021.

Yang Liu and Mirella Lapata. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75, 2018.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Guoshun Nan, Zhijiang Guo, Ivan Sekulić, and Wei Lu. Reasoning with latent structure refinement for document-level relation extraction. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 1546–1557, 2020.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4885–4901, 2020.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, 2023.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.

Panupong Pasupat and Percy Liang. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23–32, 2016.

Yu Pei, Yongping Du, and Xingnan Jin. Fover: First-order logic verification for natural language reasoning. *Transactions of the Association for Computational Linguistics*, 13:1340–1359, 2025.

Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1267–1273, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1128. URL https://aclanthology.org/N19-1128/.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8372–8388, 2020.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJzSgnRcKX.

Adina Williams, Andrew Drozdov*, and Samuel R Bowman. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267, 2018a.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, 2018b.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *Advances in neural information processing systems*, 35:32353–32368, 2022.

Kaiyu Yang, Jia Deng, and Danqi Chen. Generating natural language proofs with verifier-guided search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 89–105, 2022.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. Pcfgs can do better: Inducing probabilistic context-free grammars with many symbols. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1487–1498, 2021.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 754–765, 2018.

Xin Zhang and Victor S Sheng. Token-context attention for nli: An alternative to self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 34836–34844, 2026.

A Auxiliary Training Losses

Let $S = |\mathcal{S}|$ and $\mathcal{A} = \mathcal{S} \times \{1, \dots, K_{\text{ctx}}\}$ denote the number of context scales and the slot–scale node set, respectively. The scale-specific slot-attention parameters are shared across the L_{slot} slot layers. At slot layer ℓ, the query and key projections are $q_{k,\ell}^{(s)} = W_Q^{(s)} \, \text{LN}_Q^{(s)} \left(s_{k,\ell-1}^{(s)} \right) + b_Q^{(s)}$ and $k_{i,\ell}^{(s)} = W_K^{(s)} \, \text{LN}_{KV}^{(s)} \left(c_{i,\ell-1}^{(s)} \right) + b_K^{(s)}$, respectively. After L_{slot} layers, the resulting slot state is $\tilde{s}_{k,L_{\text{slot}}}^{(s)}$.

For node $a = (s, k)$, define the initial projected node state as $u_a^{(0)} = f_{\text{in}} \left(\tilde{s}_{k,L_{\text{slot}}}^{(s)} \right)$. For nodes a and a', the initial continuous edge state is $e_{a,a'}^{(0)} = \text{LN}_e \left(f_e \left([u_a^{(0)}; u_{a'}^{(0)}; u_a^{(0)} \odot u_{a'}^{(0)}; u_a^{(0)} - u_{a'}^{(0)}] \right) \right)$. The relational graph jointly refines the node and edge states: $(\{\hat{u}_a\}_{a \in \mathcal{A}}, \{\hat{e}_{a,a'}\}_{a,a' \in \mathcal{A}}) = \text{RelGraph} \left(\{u_a^{(0)}\}_{a \in \mathcal{A}}, \{e_{a,a'}^{(0)}\}_{a,a' \in \mathcal{A}} \right)$. The refined slot representation is obtained through an output projection and a residual connection: $\hat{s}_k^{(s)} = \tilde{s}_{k,L_{\text{slot}}}^{(s)} + f_{\text{out}} \left(\hat{u}_{(s,k)} \right)$. The continuous edge states

provide additive biases for relational attention and are updated jointly with the node states. They are not mapped to discrete edge categories or directly regularized in the reported experiments.

Let $\bar{\eta}^{(s)}_{b,k,i,\ell}$ denote the head-averaged post-dropout slot-to-context attention coefficient for mini-batch element b, slot k, valid context position i, slot layer ℓ, and scale s. The slot-attention regularization term is $\mathcal{L}_{\text{slot-attn}} = -\frac{1}{BL_{\text{slot}}SK_{\text{ctx}}} \sum_{b=1}^{B} \sum_{\ell=1}^{L_{\text{slot}}} \sum_{s\in\mathcal{S}} \sum_{k=1}^{K_{\text{ctx}}} \times \sum_{i\in\mathcal{I}(x_b)} \bar{\eta}^{(s)}_{b,k,i,\ell} \log\left(\max\left(\bar{\eta}^{(s)}_{b,k,i,\ell}, \epsilon\right)\right)$.

Let $N_{\text{slot}} = SK_{\text{ctx}}$, and let $U_{b,\ell} \in \mathbb{R}^{N_{\text{slot}}\times D}$ contain the row-wise normalized slot–scale states after slot layer ℓ. The slot-diversity term is $\mathcal{L}_{\text{slot-div}} = \frac{1}{BL_{\text{slot}}N^2_{\text{slot}}} \sum_{b=1}^{B} \sum_{\ell=1}^{L_{\text{slot}}} \left\| U_{b,\ell}U^{\top}_{b,\ell} - I_{N_{\text{slot}}} \right\|^2_F$. This term includes the diagonal entries of the Gram matrix.

Let I denote the set of valid occurrences in the mini-batch, and let $g_i = x_i$ denote the token-ID group identifier of occurrence i. Let $q_i = (q_{i,(s,k)})_{s\in\mathcal{S},\, 1\le k\le K_{\text{ctx}}}$ denote its assignment vector over the refined slot–scale nodes. The component corresponding to node (s,k) is $q_{i,(s,k)} = \frac{b^{(s),\text{rel}}_{i,k}}{\sum_{s'\in\mathcal{S}} \sum_{k'=1}^{K_{\text{ctx}}} b^{(s'),\text{rel}}_{i,k'}}$. Because the assignments within each scale sum to one, the denominator equals S.

For group identifier v, define $I_v = \{i \in I : g_i = v\}$, $\bar{q}_v = |I_v|^{-1} \sum_{i\in I_v} q_i$, and $\bar{q} = |I|^{-1} \sum_{i\in I} q_i$. The assignment losses are $\mathcal{L}_{\text{group}} = |I|^{-1} \sum_{i\in I} \mathrm{KL}\left(q_i \| \bar{q}_{g_i}\right)$, $\mathcal{L}_{\text{sharp}} = |I|^{-1} \sum_{i\in I} H(q_i)$, and $\mathcal{L}_{\text{balance}} = \mathrm{KL}\left(\bar{q} \| \,\mathrm{Unif}_{SK_{\text{ctx}}}\right)$, where $H(p) = -\sum_a p_a \log p_a$.

The slot-attention regularization coefficient follows $\lambda_{\text{ent}}(e) = 0.025\,[1 - \cos(\pi \min(e/8, 1))]$, where e is the zero-based training-epoch index.

B Dataset splits

Table 7: Dataset splits for logical-inference experiments.

Dataset	Train	Validation	Test / reporting
FOLIO	Original train excluding validation subset	Story-disjoint subset covering approximately 20% of the original training examples	Original validation set used as test set
JustLogic-mixed	70% of the mixed-depth pool, label-stratified	15% of the mixed-depth pool, label-stratified	15% of the mixed-depth pool; independently trained mixed-setting run
JustLogic-bucketed	70% of the bucket-oriented pool, label-stratified	15% of the bucket-oriented pool, label-stratified	15% of the pool; independently trained and reported for depths 1–3, 4–5, and 6–7
ProofWriter-mixed	20% of the mixed-depth training split, label-stratified	Full mixed-depth validation split	Full mixed-depth test split; independently trained mixed-setting run
ProofWriter-depth	20% of the D5 training split, label-stratified	Full D5 validation split	Full D5 test split; independently trained and reported for exact depths 0–5

Table 8: Dataset splits for natural-language-inference experiments.

Dataset	Train	Validation	Test / reporting
ANLI-R1	Original training set	Original validation set	Original test set
SNLI	5% of the training split, label-stratified	10% of the validation split, label-stratified	Complete test split
MNLI	5% of the training split, label-stratified	10% of the matched development split, label-stratified	Complete mismatched development split

C Additional Latent-Structure Diagnostics

Table 9 reports descriptive statistics of the induced latent structures in the examined FOLIO run.

Table 9: Descriptive latent-structure statistics on FOLIO.

Metric	Value
Examples with at least one token-group node	203/203
Mean token-ID groups per example	40.82
Mean off-diagonal affinity density	0.408
Mean top-two latent-state logit margin	0.852