STRUCTURE-INTERNALIZED RULE LANGUAGE MODEL FOR FAITHFUL KNOWLEDGE GRAPH REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge Graph Reasoning (KGR) aims to discover latent facts by leveraging the structural evidence available in KGs, posing a challenge to the structural semantic understanding capability of KGR models. Recent studies have demonstrated that Large Language Models (LLMs) can achieve remarkable progress on KGR tasks via flexible in-context learning. However, the inherent representation inconsistency between KG structural context and LLM parametric knowledge remains inadequately addressed. This limitation prevents LLMs from effectively perceiving reasoning evidence that aligns with KG constraints, which undermines both the effectiveness and faithfulness of reasoning. We refer to this problem as *reasoning evidence perception drift* of LLMs over KGs. To address this problem, we propose a Structure-Internalized Rule Language Model (SIRLM), which centers on structural rule generation to couple the parametric learning of structural knowledge with the faithfulness evaluation of reasoning logic, enabling LLMs to anchor tightly to KG-grounded evidence. Specifically, we first design a Structure-Internalized Rule Generator (SIRG), which incorporates an in-context learning block augmented with a structural relation memory to coordinate structural and parametric knowledge. Furthermore, we equip SIRG with a KG tokenizer based on structural invariance learning and a neuro-symbolic reasoner based on rule-constrained message propagation. These components provide SIRG with learnable structural representations and faithful rule-execution feedback, respectively. Our SIRLM can be seamlessly integrated into standard LLM training paradigms, such as SFT and GRPO. Extensive experiments against 17 state-of-the-art KGR methods on 36 datasets demonstrate the significant superiority of SIRLM[1].

1 INTRODUCTION

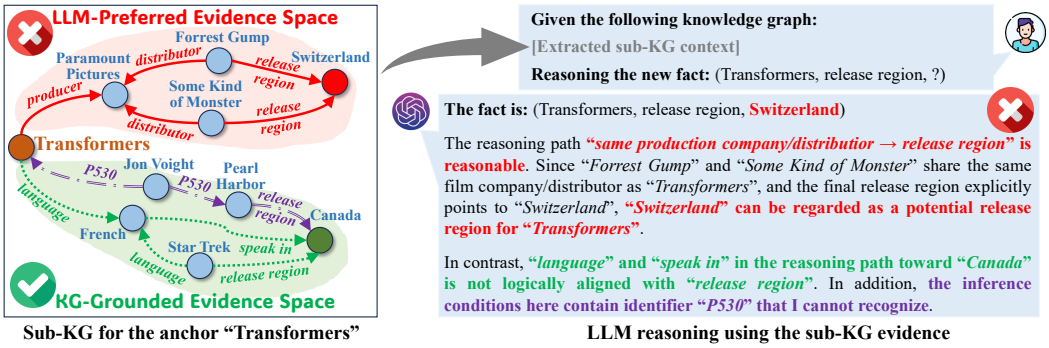
Knowledge Graphs (KGs) (Ji et al., 2022) represent real-world facts through structural triplets composed of entities and relations, maintaining a unified framework for storing and querying large-scale knowledge. However, KGs are generally incomplete, which leaves many potential facts unobserved. Consequently, Knowledge Graph Reasoning (KGR) (Liang et al., 2024) has been proposed to leverage the available KG context to infer missing facts, which provides more sufficient evidence for knowledge-driven applications (Huang et al., 2026; Luo et al., 2024).

Traditional KGR methods mainly focus on knowledge embedding (Sun et al., 2019) and logical rule learning (Sadeghian et al., 2019), which achieve efficient fact prediction via structural representation learning and interpretable reasoning (Wan & Du, 2021), respectively. However, these methods generally rely on specific contextual instances within a KG. When confronted with an unfamiliar KG, they often require complex rule mining or model retraining, which makes it difficult for these methods to generalize to more challenging KGR scenarios (Zhu et al., 2022; Lee et al., 2023).

Recently, Large Language Models (LLMs) (Naveed et al., 2025), pre-trained on massive natural language corpora, have demonstrated outstanding performance in various reasoning tasks. Latest studies have confirmed that LLMs can achieve breakthrough success on KGR tasks (Kim et al., 2023; Wang et al., 2024a; Zhuo et al., 2025b). With the support of flexible in-context learning capabilities, LLMs integrate inherent parametric knowledge with structural context of KGs and exhibit remarkable knowledge emergence (Pan et al., 2024), thereby discovering new facts in KGs that are not explicitly

[1]Our source codes are available at https://anonymous.4open.science/r/SIRLM-B35B/

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.



Sub-KG for the anchor "Transformers" **LLM reasoning using the sub-KG evidence**

Figure 1: Case of reasoning evidence perception drift of LLMs over KGs. LLM select incorrect reasoning evidence based on its semantic preference, resulting in its inability to follow KG-grounded reasoning logic to arrive at correct reasoning conclusions.

observed. More importantly, compared with traditional KGR methods, LLM-based approaches show stronger transferability when facing previously unseen KG structures in inductive reasoning settings (Wang et al., 2024b; Guo et al., 2024; Zhuo et al., 2026).

Despite significant accomplishments, existing LLM-based KGR methods still struggle to address the representation inconsistency between structural and parametric knowledge (Jiang et al., 2024b), which constrains the ability of LLMs to grasp structural KG context during reasoning. This limitation is reflected in the fact that most LLMs tend to select reasoning evidence from a KG based on their pre-trained semantic preferences, rather than strictly following the structural constraints induced from the KG. As a result, LLMs are inclined to generate reasoning logic that appears semantically plausible but is difficult to ground in a KG, leading to the reasoning bias and the faithfulness degradation of results. We refer to this issue as the ***reasoning evidence perception drift*** problem of LLMs over KGs, meaning that inconsistencies in knowledge representation hinder LLMs from identifying reasonable evidence from KGs, thereby limiting the effectiveness of LLM reasoning.

Figure 1 illustrates the harm that such a perception shift brings to LLMs. In this case, LLMs tend to accept the assertion "*films produced or distributed by the same company are likely to be released in the same region*" based on their semantic preferences. As a result, the evidence space is anchored to the sub-KG marked in **red** in Figure 1. However, the structural rule that can be explicitly induced from the KG, namely $release_region(X, Y) = language(X, Z) \wedge speak_in(X, Z)$, is instead misjudged as a weakly related evidence space (marked in **green** in Figure 1). In addition, for relation paths represented by anonymized identifiers (marked in **purple** in Figure 1), the absence of natural language semantics often leads LLMs to deliberately ignore this structural evidence that could potentially support correct reasoning. This preference for pre-trained semantic cues and the neglect of structural context cause the LLM's reasoning process to deviate from the correct KG-grounded evidence space, ultimately producing erroneous results.

To address the aforementioned limitation, we propose a Structure-Internalized Rule Language Model (SIRLM), which enables LLMs to accurately perceive KG-grounded evidence for faithful reasoning. Overall, SIRLM consists of a KG tokenizer, a Structure-Internalized Rule Generator (SIRG) and a neuro-symbolic reasoner. In this framework, SIRG serves as the core module to generate structurally grounded rules. Therefore, we propose an in-context learning block based on a Structural Relation Memory (SRM) mechanism, which allows SIRG to align pre-trained knowledge with structural context during rule generation. Complementing SIRG, the KG tokenizer is built upon the Structural Invariance Learning (SIL) principle (Galkin et al., 2024; Huang et al., 2025a; Zhang et al., 2024b), which encodes KG elements into universal structure representations, enabling LLMs to grasp them in the form of new tokens without specific textual annotations. Furthermore, we design the neuro-symbolic reasoner as a rule executor based on Rule-Constraint Message Propagation (RCMP), which feeds back a rule faithfulness metric composed of reasoning conclusions to SIRG, forming a closed-loop optimization that reinforces structurally consistent reasoning.

In terms of model training, SIRLM can seamlessly integrate with standard Supervised Fine-Tuning (SFT) and post-training frameworks of LLMs. This capability endows SIRLM with significant transferable advantages across a wide range of cross-scenario KGR tasks.

Our main contributions can be summarized as follows:

- We propose a structure-internalized rule language model to mitigate the reasoning evidence perception drift of LLMs over KGs, enabling reliable LLM-based KG reasoning.
- We design a structure-internalized rule generator with a structural relation memory to coordinate LLM knowledge with KG context, together with a SIL-based KG tokenizer and a RCMP-based rule reasoner for universal structure representation learning and faithful rule execution feedback.
- SIRLM can seamlessly integrate with standard LLM training frameworks such as SFT and GRPO, achieving generalizable reasoning across unknown KGs.
- Extensive experimental results conducted on 36 datasets demonstrate that SIRLM exhibits remarkable reasoning capabilities in both transductive and inductive KGR scenarios.

2 Related Work

KGR studies have long focused on mining latent facts from KGs, providing interpretable knowledge support for downstream tasks. Traditional KGR methods adopt structural representation learning techniques by projecting entities and relations into appropriate embedding spaces, such as Euclidean (Bordes et al., 2013), complex (Trouillon et al., 2016), or manifold spaces (Xiao et al., 2016), to infer latent associations between entities and relations. To ensure interpretability in reasoning, some studies introduce rule-based approaches that leverage techniques such as rule mining (Qu et al., 2021; Sadeghian et al., 2019), relational path retrieval (Das et al., 2018), or Markov decision processes (Lao & Cohen, 2010) to perform KGR while providing explicit reasoning paths.

As KGs scale up, early embedding-based and rule-based methods struggle to adapt to more complex knowledge structures. Consequently, researchers have incorporated Graph Neural Networks (GNN) into KGR (Galkin et al., 2022; Teru et al., 2020; Zhu et al., 2022). Methods such as NBFNet (Zhu et al., 2021), RED-GNN (Zhang & Yao, 2022), and InGram (Lee et al., 2023) exploit the message-passing mechanism of GNNs to capture fine-grained structural context about entities and relations, providing stronger evidence for comprehensive reasoning. Building on this, some studies (Galkin et al., 2024; Huang et al., 2025a; Zhang et al., 2024b; Cui et al., 2024) focus on improving the generalization of KGR models to out-of-distribution graph structures. These approaches aim to discover reusable minimal subgraph patterns within KGs and learn universal structural representations of entities and relations, *i.e.*, Structural Invariance Learning (SIL) (Galkin et al., 2024). This technique enables unknown KG elements to be projected into learned structural patterns, thereby enhancing model performance in complex reasoning scenarios.

In recent years, LLM-based KGR has emerged as a new research focus (Wang et al., 2024b; Zhuo et al., 2025a; 2026). By leveraging the strong knowledge emergence and in-context learning capabilities of LLMs (Pan et al., 2024), these approaches uncover deeper facts in KGs. For example, methods such as KICGPT (Wei et al., 2023) and ChatRule (Luo et al., 2025) adopt the *LLM-planning and KG-retrieval* paradigm to prompt LLMs in performing structural reasoning over given sub-KGs. Meanwhile, approaches such as KoPA (Zhang et al., 2024a), MKGL (Guo et al., 2024), and KRLM (Zhuo et al., 2026) rely on fine-tuning techniques to inject structural knowledge into the pre-trained space of LLMs, aligning their parametric knowledge with the structural knowledge of KGs. Despite significant development, existing LLM-based methods still face the challenge in handling the widely observed KG-LLM representation gaps. Under this condition, structural knowledge that is beneficial for reasoning in KGs is often regarded by LLMs as irrelevant evidence, leading to a phenomenon known as *evidence perception drift*.

The aforementioned challenge have prompted us to propose a more refined mechanism for internalizing structural knowledge, which uses structural rule generation as a feedback to guide LLMs in perceiving grounded structural logic over KGs, thereby improving model reasoning.

3 Preliminaries

In this section, we introduce the background and key conceptions of our study.

3.1 Knowledge Graph Reasoning

We define a KG as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively. $\mathcal{T} = \{< e_h, r_q, e_t > | e_h, e_t \in \mathcal{E}, r_q \in \mathcal{R}\}$ is the set of triples. Each triple represents a factual statement indicating that the head entity e_h and the tail entity e_t are connected via relation r_q.

KGR aims to infer new facts of the form $< e_h, r_q, ? > \notin \mathcal{T}$. Given a training KG $\mathcal{G}_{tr} = (\mathcal{E}_{tr}, \mathcal{R}_{tr}, \mathcal{T}_{tr})$ and a inference KG $\mathcal{G}_{inf} = (\mathcal{E}_{inf}, \mathcal{R}_{inf}, \mathcal{T}_{inf})$, KGR tasks are typically categorized into **transductive** ($\mathcal{E}_{tr} = \mathcal{E}_{inf}$, $\mathcal{R}_{tr} = \mathcal{R}_{inf}$, and $\mathcal{T}_{tr} \neq \mathcal{T}_{inf}$) and **inductive** setting ($\mathcal{E}_{tr} \neq \mathcal{E}_{inf}$ or $\mathcal{R}_{tr} \neq \mathcal{R}_{inf}$, and $\mathcal{T}_{tr} \neq \mathcal{T}_{inf}$). The inductive setting requires KGR models to generalize to previously unseen entities or relations, thereby ensuring strong extrapolation and generalization capabilities.

3.2 Structural Invariance Learning of KG

SIL of KG derives universal structural representations for unseen entities and relations through a relational graph and a Neural Bellman-Ford Network (NBFNet) (Zhu et al., 2021).

The relational graph is designed to characterize the structural motifs between relations in a KG. Let the relational graph be defined as $\mathcal{G}_r = (\mathcal{R}, \mathcal{R}^*, \mathcal{T}^*)$, where $\mathcal{R}$ corresponds to the set of relations in the original KG $\mathcal{G}$ and serves as the node set of $\mathcal{G}_r$. $\mathcal{R}^*$ denotes the set of motif edges to connect relation nodes. The detailed definition of motif edges is provided in **Appendix B**.

Under this inductive paradigm, any relation in an arbitrary KG can be mapped into $\mathcal{G}_r$. By aggregating these motif edges, we can obtain the universal structural representation for any relation.

NBFNet is a GNN that generalizes path formulations over graph nodes. Given a query triple $< e_h, r_q, ? >$, we first perform NBFNet$(U, \boldsymbol{R}^*, \mathcal{G}_r)$, an N-layer NBFNet, over $\mathcal{G}_r$:

$$\text{NBFNet}(U, \boldsymbol{R}^*, \mathcal{G}_r) = \begin{cases} \boldsymbol{r}_{j|q}^{(0)} = \textsc{Init}(r_j, U), \quad \text{where } U = \{< r_q, \mathbf{1}^d >\}, r_q, r_j \in \mathcal{R}, \\ \boldsymbol{r}_{j|q}^{(n)} = \textsc{Up}\Big(\boldsymbol{r}_{j|q}^{(n-1)}, \textsc{Agg}\big(\{\textsc{Msg}(\boldsymbol{r}_{z|q}^{(n-1)}, \boldsymbol{r}^*) | r_z \in \mathcal{N}_{r^*}(r_j)\} | r^* \in \mathcal{R}^*, \boldsymbol{r}^* \in \boldsymbol{R}^*\big)\Big) \end{cases} \tag{1}$$

where U is a tuple set that stores conditions that satisfy non-zero initialization nodes, $\mathbf{1}^d$ is a d-dimensional full-one vector, $n \in [1, N]$ is the layer index, and $\boldsymbol{R}^* \in \mathbb{R}^{|\mathcal{R}^*| \times d}$ is the randomly initialized embeddings of motif edges. $\textsc{Init}(\cdot)$, $\textsc{Msg}(\cdot)$, $\textsc{Agg}(\cdot)$, and $\textsc{Up}(\cdot)$ are an indicator function, the DistMult message propagation (Yang et al., 2015), the summation aggregation operation, and a Multi-Layer Perceptron (MLP), respectively. The details of Eq. (1) is provided in Eq. (17).

According to Eq. (1), we obtain the structural representations of all relations conditioned on r_q, denoted as $\boldsymbol{R}_{|q} = \text{NBFNet}(U, \boldsymbol{R}^*, \mathcal{G}_r)$. Based on this, we further compute the structural representations of all entities conditioned on e_h as $\boldsymbol{E}_{|h}$:

$$\boldsymbol{E}_{|h} = \text{NBFNet}(\{< e_h, \boldsymbol{r}_{q|q}^{(N)} >\}, \boldsymbol{R}_{|q}, \mathcal{G}), \quad \boldsymbol{r}_{q|q}^{(N)} \in \boldsymbol{R}_{|q}. \tag{2}$$

The details of Eq. (2) is provided in Eq. (18).

Finally, we compute scores for the candidate tail entities of the query triple $< e_h, r_q, ? >$:

$$s_{\text{KG}}^{(i)} = \mathcal{S}_{\text{KG}}(\boldsymbol{e}_{i|h}^{(N)} || \boldsymbol{r}_{q|q}^{(N)}) \quad i \in [1, |\mathcal{E}|], \tag{3}$$

where $\boldsymbol{e}_{i|h} \in \boldsymbol{E}_{|h}$ and $\mathcal{S}_{\text{KG}} : \mathbb{R}^{2d} \to \mathbb{R}^1$ is a scoring function.

3.3 In-context learning and next-token prediction of LLM

In context learning: Let X be an instruction and TKN_{LLM} denote the pre-trained token embedding table of LLM. By looking up X in TKN_{LLM}, the instruction is transformed into a sequence $\text{TKN}_{\text{LLM}}[X] = \boldsymbol{X} \in \mathbb{R}^{L \times F}$ with L F-dimensional token embeddings. The LLM aggregates each token embedding $\boldsymbol{x}_i \in \boldsymbol{X}$ together with the embeddings of all preceding tokens through the self-attention mechanism to perform in-context learning, thereby producing the hidden state of $\boldsymbol{x}_i$, defined as $\boldsymbol{h}_i = \text{InCon}(\boldsymbol{x}_i, \boldsymbol{X}_{\leq i})$:

$$\text{InCon}(\boldsymbol{x}_i, \boldsymbol{X}_{\leq i}) = \text{FFN}\Big(\text{softmax}\big(\frac{f_Q(\boldsymbol{x}_i)[f_K(\boldsymbol{X}_{\leq i})]^{\text{T}}}{\sqrt{F}}\big) f_V(\boldsymbol{X}_{\leq i})\Big), \tag{4}$$

where $f_{\{Q,K,V\}}(\cdot) : \mathbb{R}^F \to \mathbb{R}^F$ are pre-trained linear layers and $\text{FFN}(\cdot) : \mathbb{R}^F \to \mathbb{R}^F$ is a pre-trained Feed Forward Network (FFN).

Next-token prediction: Based on Eq. (4), the new token generated by LLM conditioned on the input instruction X can be expressed as:

$$\hat{\boldsymbol{x}}_{z+1} \leftarrow \underset{\hat{\boldsymbol{x}} \in \text{TKN}_{\text{LLM}}}{\arg\max} P_{\boldsymbol{\Theta}}(\hat{\boldsymbol{x}} | \text{InCon}(\hat{\boldsymbol{x}}_z, [\boldsymbol{X} : \hat{\boldsymbol{X}}_{\leq z}])), \tag{5}$$

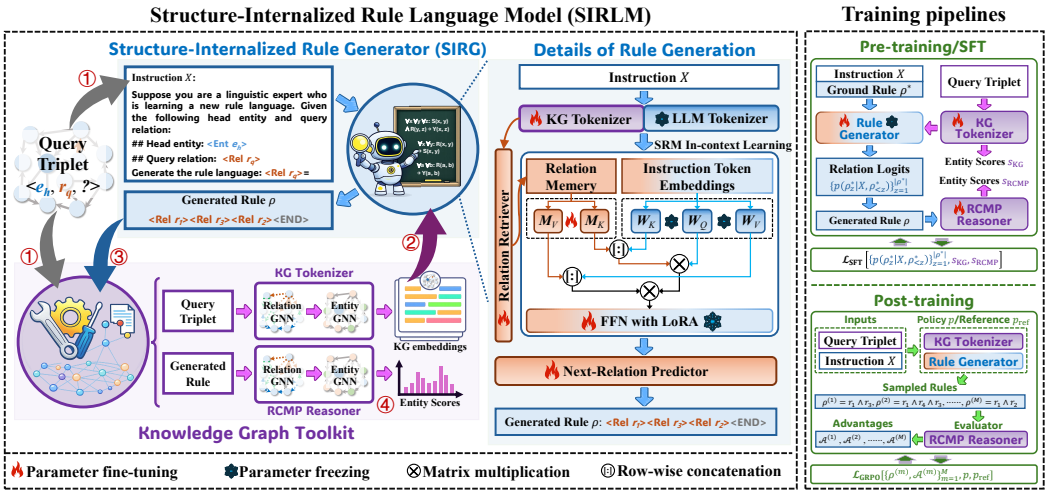


Figure 2: Overall architecture and training framework of SILRM. Given a query triplet, we first ① convert it into an instruction X and input the triplet into the KG tokenizer to obtain structural embeddings of entities and relations. These structural representations are then ② tokenized along with the LLM pre-trained tokenizer for X. Next, SIGR reads the aforementioned tokens and ③ generates structural rules for the RCMP reasoner. Afterwards, ④ the reasoner executes rules and performs neuro-symbolic reasoning, ultimately obtaining scores for candidate entities.

where $\hat{\boldsymbol{X}}_{\leq z} \in \mathbb{R}^{z \times F}$ is the embeddings of the first z generated tokens and $[:]$ denotes a row-wise concatenation operation. $\text{InCon}(\hat{\boldsymbol{x}}_z, [\boldsymbol{X} : \hat{\boldsymbol{X}}_{\leq z}])$ represents the hidden state of the z-th generated token $\hat{\boldsymbol{x}}_z \in \hat{\boldsymbol{X}}_{\leq z}$, which is projected through a MLP module parameterized by $\boldsymbol{\Theta}$ to produce the generation probability of the (z+1)-th token.

4 Methodology

In this section, we elaborate on the proposed SILRM in detail. As illustrated in Figure 2, SILRM consists of a structure-internalized rule generator and a KG toolkit. In the following, we describe our method from four aspects: the construction of the query instruction and the KG tokenizer (**Section 4.1**), LLM-based in-context learning and rule generation (**Section 4.2**), and rule reasoning (**Section 4.3**). Finally, we introduce how our framework seamlessly integrates with LLM SFT and post-training strategies in **Section 4.4**.

4.1 Construction of Query Instruction and KG Tokenizer

Given a query triplet $< e_h, r_q, ? >$, we design a query instruction X suitable for generating structural rules using LLMs, whose detailed schema is provided in **Appendix A**.

In the instruction schema, strings of the form $<$ Ent ID $>$ and $<$ Rel ID $>$ are treated as indivisible structural token identifiers, which denote specific entities and relations in a KG, respectively. $<$ END $>$ serves as a rule body termination. These structural tokens are embedded via a KG tokenizer TKN_{KG} and are jointly processed with the pre-trained LLM tokenizer TKN_{LLM} to encode the remaining text in instruction X. To unify the processing of structural and textual tokens in X, the tokenizer of the rule generator can be expressed as the combination of TKN_{KG} and TKN_{LLM}:

$$\text{TKN}_{\text{RG}} = [\text{TKN}_{\text{LLM}} : \text{TKN}_{\text{KG}}], \quad \text{TKN}_{\text{KG}} = f_{\text{up}}([\boldsymbol{r}_{\text{end}} : \boldsymbol{E}_{|h} : \boldsymbol{R}_{|q}]), \tag{6}$$

where $\boldsymbol{r}_{\text{end}} \in \mathbb{R}^d$ is a randomly initialized embedding for $<$ END $>$, $f_{\text{up}}(\cdot) : \mathbb{R}^d \to \mathbb{R}^F$ is a trainable linear layer, and $\boldsymbol{R}_{|q} \in \mathbb{R}^{|\mathcal{R}| \times d}$ and $\boldsymbol{E}_{|h} \in \mathbb{R}^{|\mathcal{E}| \times d}$ are the structural embeddings of entities and relations obtained by Eqs. (1) and (2), respectively. It is important to note that the pre-trained TKN_{LLM} remain frozen and only TKN_{KG} is trained. This allows LLMs to perceive the structural representations of the KG through newly introduced tokens.

4.2 In-context Learning and Rule Generation

To enable LLMs to generate executable rules, we design an in-context learning strategy with the SRM mechanism and subsequently perform autoregressive relation token prediction, which ensures that

the generated rules can be grounded in the corresponding KG, providing interpretable evidence for subsequent knowledge reasoning.

In-context learning with structural relation memory: To guide the LLM toward structure-internalized rule generation, we adapt the pre-trained attention layer by integrating a structural relation memory mechanism. Given a query triplet $< e_h, r_q, ? >$, we first compute a relevance score for each relation $r_j \in \mathcal{R}$ and select the top-K relation as the memory $\boldsymbol{R}_{\text{mem}}$:

$$\boldsymbol{R}_{\text{mem}} = \{\boldsymbol{r}_k \in \boldsymbol{R}_{|q} | s_{\text{rel}}^{(k)} \in \text{TopK}(\{s_{\text{rel}}^{(j)}\}_{j=1}^{|\mathcal{R}|})\}_{k=1}^{K}, \quad s_{\text{rel}}^{(j)} = \mathcal{S}_{\text{rel}}(\boldsymbol{e}_{h|h}^{(N)} || \boldsymbol{r}_{q|q}^{(N)} || \boldsymbol{r}_{j|q}^{(N)}), \tag{7}$$

Here, $\boldsymbol{e}_{h|h}^{(N)} \in \boldsymbol{E}_{|h}$; $\boldsymbol{r}_{q|q}^{(N)}, \boldsymbol{r}_{j|q}^{(N)} \in \boldsymbol{R}_{|q}$; and $\mathcal{S}_{\text{rel}}(\cdot) : \mathbb{R}^{3d} \to \mathbb{R}^d$ is a MLP scorer.

Let the instruction X be tokenized by TKN_{RG} into the embedding sequence $\boldsymbol{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^{L}$, the in-context learning operation in Eq. (4) can be improved to:

$$\text{InCon}(\boldsymbol{x}_i, \boldsymbol{R}_{\text{mem}}, \boldsymbol{X}_{\leq i}) = \text{FFN}\Big(\text{softmax}\big(\frac{f_Q(\boldsymbol{x}_i)[f_K(\boldsymbol{X}_{\leq i}) : m_K(\boldsymbol{R}_{\text{mem}})]^{\text{T}}}{\sqrt{F}}\big)[f_V(\boldsymbol{X}_{\leq i}) : m_V(\boldsymbol{R}_{\text{mem}})]\Big), \tag{8}$$

where $\text{FFN}(\cdot)$ incorporates a LoRA fine-tuning block (Hu et al., 2022) in practical modeling and $m_{\{K,V\}}(\cdot) : \mathbb{R}^d \to \mathbb{R}^F$ are trainable linear layers. Eq. (8) allows the hidden state of last token x_L to fully integrate the multi-modal token representations in X and the relation memory, providing semantically complete activation for the subsequent rule generation. Its effectiveness analysis is provided in **Appendix D**.

Next-relation prediction for rule generation: Let the hidden state of the last token be $\boldsymbol{h}_L = \text{InCon}(\boldsymbol{x}_L, \boldsymbol{R}_{\text{mem}}, \boldsymbol{X})$, we formulate rule generation as an autoregressive next-relation prediction process according to Eq. (5). First, the distribution $P_{\boldsymbol{\Theta}}$ in Eq. (5) can be concretized as:

$$P_{\boldsymbol{\Theta}}(\boldsymbol{r}|\boldsymbol{h}_L) = f_{\boldsymbol{\Theta}}(\boldsymbol{h}_L || \boldsymbol{r}), \quad \boldsymbol{r} \in [\boldsymbol{r}_{\text{end}} : \boldsymbol{R}_{|q}], \tag{9}$$

where $f_{\boldsymbol{\Theta}}(\cdot) : \mathbb{R}^{F+d} \to \mathbb{R}^1$ is a trainable MLP module, which scores each candidate relation embedding to determine its likelihood as the next generated token.

Then, relation tokens are generated sequentially in an autoregressive manner:

$$\boldsymbol{r}_{z+1} \leftarrow \underset{\boldsymbol{r} \in [\boldsymbol{r}_{\text{end}} : \boldsymbol{R}_{|q}]}{\arg\max} P_{\boldsymbol{\Theta}}(\boldsymbol{r}|\boldsymbol{h}_{L+z}), \quad \boldsymbol{h}_{L+z} = \text{InCon}\Big(f_{\text{up}}(\boldsymbol{r}_z), \boldsymbol{R}_{\text{mem}}, [\boldsymbol{X} : f_{\text{up}}(\boldsymbol{R}_{\leq z})]\Big), \tag{10}$$

where $z \in [0, \varepsilon - 1]$, ε denotes the maximum rule length, $\boldsymbol{R}_{\leq z}$ consists of the structural embeddings of the previously generated z relations, and $f_{\text{up}}(\cdot) : \mathbb{R}^d \to \mathbb{R}^F$ is a linear layer defined in Eq. (6). The generation process terminates when $\boldsymbol{r}_{z+1} = \boldsymbol{r}_{\text{end}}$ or $z + 1 = \varepsilon$.

Through Eq. (10), we can obtain the rule body $\rho = \bigwedge_{z=1}^{\varepsilon} r_z$ together with their hidden states $\{\boldsymbol{h}_{L+z}\}_{z=1}^{\varepsilon}$, which are used for subsequent rule reasoning for $< e_h, r_q, ? >$.

4.3 Reasoning with Rule-constraint Message Propagation

Unlike conventional SIL methods that initialize relational graph nodes solely with a full-one vector of the query relation, our RCMP mechanism incorporates the hidden states $\{\boldsymbol{h}_{L+z}\}_{z=1}^{\varepsilon}$ of the rule path $\bigwedge_{z=1}^{\varepsilon} r_z$ generated in Eq. (10) as additional initialization signals. Specifically, we modify the initialization step of relation nodes in Eq. (1) as follows:

$$\hat{\boldsymbol{R}}_{|q} = \text{NBFNet}(\{< r_q, \mathbf{1}^d >\} \cup \{< r_z, f_{\text{down}}(\boldsymbol{h}_{L+z}) >\}_{z=1}^{\varepsilon}, \hat{\boldsymbol{R}}^*), \tag{11}$$

where $f_{\text{down}}(\cdot) : \mathbb{R}^F \to \mathbb{R}^d$ is a trainable linear layer. Similar to Eq. (1), $\hat{\boldsymbol{R}}^*$ is a randomly initialized embeddings of motif edges. Eq. (11) transforms the implicit semantic signals of the rule sequence into explicit structural constraints, which enhances the contextual representation and structural discriminability of the query relation. The details of Eq. (11) is provided in Eq. (19) and a theoretical analysis of RCMP is provided in **Appendix E**.

Based on Eq. (11), we obtain $\hat{\boldsymbol{R}}_{|q}$ as the relation representations. Then, we compute entity representations $\hat{\boldsymbol{E}}_{|h}$ and score the candidate tail entities of $< e_h, r_q, ? >$:

$$\hat{\boldsymbol{E}}_{|h} = \text{NBFNet}(\{< e_h, \hat{\boldsymbol{r}}_{q|q}^{(N)} >\}, \hat{\boldsymbol{R}}_{|q}, \mathcal{G}), \tag{12}$$

$$s_{\text{RCMP}}^{(i)} = \mathcal{S}_{\text{RCMP}}(\hat{\boldsymbol{e}}_{i|h}^{(N)} || \hat{\boldsymbol{r}}_{q|q}^{(N)}), \hat{\boldsymbol{e}}_{i|h} \in \hat{\boldsymbol{E}}_{|h}^{(N)}, \hat{\boldsymbol{r}}_{q|q}^{(N)} \in \hat{\boldsymbol{R}}_{|q}, \tag{13}$$

where $\mathcal{S}_{\text{RCMP}}(\cdot) : \mathbb{R}^{2d} \to \mathbb{R}^1$ is a MLP scorer. The details of Eq. (12) is provided in Eq. (20).

4.4 TRAINING FRAMEWORKS

During the pre-training or SFT phase, SIRLM incorporates the KG tokenizer and RCMP reasoner into the standard LLM training framework with supervision signals. Let ρ^* and e_t represent the ground truth rule and target entity of $< e_h, r_q, ? >$, respectively. The pre-training/SFT loss for SIRLM can be expressed as:

$$\mathcal{L}_{\text{SFT}} = -\Big[\sum_{z=1}^{|\rho^*|} \log\big(p(\rho_z^*|X, \rho_{<z}^*)\big) + \log\big(sc_{\text{KG}}^{(t)} sc_{\text{RCMP}}^{(t)}\big) - \frac{1}{|\mathcal{N}|}\sum_{e_i \in \mathcal{N}} \log\big((1 - sc_{\text{KG}}^{(i)})(1 - sc_{\text{RCMP}}^{(i)})\big)\Big], \quad (14)$$

where $p(\rho_z^*|X, \rho_{<z}^*)$ represents the generation probability of the z-th relation token given the instruction X and the first z-1 generated relation tokens. $sc_{\text{KG}}^{(t)}$ and $sc_{\text{RCMP}}^{(t)}$ are calculated by Eqs. (7) and (13), respectively, and $\mathcal{N}$ is the negative target set of $< e_h, r_q, ? >$.

SIRLM is compatible with the post-training paradigm of existing LLMs. When transferring the pre-trained SIRLM on a downstream KG, we use the GRPO framework to train SIRLM:

$$\mathcal{L}_{\text{GRPO}} = -\frac{1}{\sum_{m=1}^{M}|\rho^{(m)}|}\sum_{z=1}^{|\rho^{(m)}|}\Big[p(\rho_z^{(m)}|X, \rho_{<z}^{(m)})\mathcal{A}^{(m)} - \beta\text{KL}\big(p(\rho_z^{(m)}|X, \rho_{<z}^{(m)})|p_{\text{ref}}(\rho_z^{(m)}|X, \rho_{<z}^{(m)})\big)\Big], \quad (15)$$

where p and p_{ref} represent the policy model and the reference model initialized by the pre-trained SIRLM, respectively. $\{\rho^{(m)}\}_{m=1}^{M}$ denotes the M rules generated by SIRLM for $< e_h, r_q, ? >$ through a sampling strategy and β is a fixed weight. The advantage $\mathcal{A}^{(m)} = \frac{\mathcal{W}^{(m)} - \text{mean}(\{\mathcal{W}^{(m)}\}_{m=1}^{M})}{\text{std}(\{\mathcal{W}^{(m)}\}_{m=1}^{M})}$ of each rule is represented as the standardization of the corresponding reward $\mathcal{W}^{(m)}$:

$$\mathcal{W}^{(m)} = \frac{\mathbb{I}\big(\phi(\rho^{(m)})\big)}{\text{Rank}\big(sc_{\text{RCMP}}^{(t)}\big)} - \mathbb{I}\big(\neg\phi(\rho^{(m)})\big), \quad \phi(\rho) = \begin{cases}\text{True}, \rho_{<|\rho|} \in \mathcal{R} \text{ and } \rho_{|\rho|} =< \text{END} > \\ \text{False, else}\end{cases}. \quad (16)$$

Here, $\text{Rank}\big(sc_{\text{RCMP}}^{(t)}\big)$ denotes the score ranking of e_t among all entities when $< e_h, r_q, ? >$ and the generated rule $\rho^{(m)}$ are given, where the score $sc_{\text{RCMP}}^{(t)}$ is obtained by Eq. (13). $\phi(\cdot)$ is a boolean function used to examine the structure compliance of the generated rule. It should be noted that at this time, the RCMP reasoner, as a evaluation model in GRPO, does not participate in the parameter optimization of Eq. (15).

5 EXPERIMENTS

In this section, we demonstrate SIRLM from the following research question: **RQ1**. Can SIRLM achieve significant performance across a wide range of transductive and inductive KGR scenarios? **RQ2**. Does the core modules of SIRLM play a crucial role in effect enhancement, including the Multi-Modal Query Instrction (MMQI) containing textual and structural tokens, the SRM mechanism in the SIRG module, and the RCMP reasoner? **RQ3**. Is SIRLM sensitive to hyperparameter settings, including the number of layers N of NBFNet, the size K of structural relation memory, and the number of rule samples M of GRPO? **RQ4**. Is SIRLM applicable to different LLM backbones?

5.1 DATASETS, BASELINES, AND EXPERIMENTAL SETTINGS

Datasets. To comprehensively evaluate SIRLM, we conduct experiments on 36 KGR datasets including four **Transductive** datasets (FB15k-237 (Toutanova & Chen, 2015), WN18RR (Dettmers et al., 2018), CoDEx-M (Safavi & Koutra, 2020), and NELL995 (Xiong et al., 2017)), 12 **Ind**uctive **E**ntity (**IndE**) datasets (Teru et al., 2020) created from FB15k237, WN18RR, and NELL995, and 20 **Fully Ind**uctive (**FullInd**) datasets created from FB15k237, NELL995, Wikidata68K (Gesese et al., 2022), and MTDEA (Zhou et al., 2023). Detailed dataset descriptions are provided in **Appendix H**.

We perform SIRLM on the aforementioned datasets using four paradigms: End-to-End (E2E) training from scratch, Pre-Training (PT), SFT, and GRPO post-training. The training settings for each paradigm are detailed in **Appendix H**.

Baselines. We compare SIRLM with **(1) conventional KG embedding methods** (TransE (Bordes et al., 2013), RotatE (Sun et al., 2019), and TuckER (Balazevic et al., 2019)), **(2) rule-based methods** (NeuralLP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), and RNNLogic (Qu et al., 2021)), **(3)**

Table 1: The overall performance of various methods on different datasets, where the MRR and Hit10 in the IndE and FullInd scenarios are summarized as average values. The colored cells represent the best, second-best, and third-best values, respectively. "–" indicates that the experimental results are unavailable, and "NA" indicates that a model is not applicable to a KGR task.

Type	Methods	FB15k237		CoDEx-M		WN18RR		NELL995		12 IndE Datasets		20 FullInd Datasets	
		MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑
Embedding methods	TransE	0.313	0.495	0.320	0.481	0.226	0.501	0.401	0.501	NA	NA	NA	NA
	RotatE	0.338	0.533	0.325	0.466	0.476	0.571	0.483	0.565	NA	NA	NA	NA
	TuckER	0.358	0.544	0.328	0.458	0.470	0.526	0.520	0.624	NA	NA	NA	NA
Rule-based methods	NeuralLP	0.237	0.362	0.334	0.444	0.435	0.566	0.394	0.482	0.449	0.623	NA	NA
	DRUM	0.343	0.516	0.312	0.438	0.486	0.586	0.532	**0.662**	0.458	0.621	NA	NA
	RNNLogic	0.344	0.530	0.310	0.445	0.483	0.558	0.416	0.478	-	-	NA	NA
GNN-based methods	NBFNet	**0.415**	**0.599**	0.343	0.509	0.551	**0.666**	0.525	0.639	0.527	0.670	0.122	0.259
	RED-GNN	0.374	0.558	0.342	0.499	0.533	0.624	**0.543**	0.651	0.504	0.648	0.151●	0.242●
	ULTRA	0.368	0.564	0.372	0.525	0.480	0.614	0.509	**0.660**	0.565	0.724	0.366	0.529
	MOTIF	0.357	0.550	0.361	0.517	0.529	0.628	0.514	0.655	0.582	0.739	0.373	0.535
LLM-based methods	KICGPT	0.412	0.554	-	-	0.549	0.641	-	-	NA	NA	NA	NA
	LSA	0.356	0.538	-	-	0.469	0.542	0.469	0.649	-	-	-	-
	KG-FIT	0.362	0.572	-	-	**0.553**	**0.695**	-	-	NA	NA	NA	NA
	ChatRule†	0.383	0.564	0.320	0.481	0.445	0.518	0.511	0.627	0.535	0.660	0.319	0.472
	FtG	0.392	0.542	**0.395**	0.473	-	-	0.538	0.626	NA	NA	NA	NA
	MKGL†	**0.415**	0.591	0.355	0.518	**0.552**	0.656	0.530	0.649	0.577	0.721	NA	NA
	KRLM†	0.394	0.568	0.367	**0.526**	**0.552**	**0.659**	0.533	0.651	0.578	0.737	0.379	0.540
Ours	$\text{SIRLM}_{\text{E2E}}$	**0.427**	**0.599**	0.368	0.525	**0.552**	0.644	**0.541**	0.642	**0.588**	**0.741**	**0.383**	**0.545**
	SIRLM_{PT}	0.408	**0.593**	0.367	0.522	0.461	0.556	0.520	0.638	0.580	0.738	0.374	0.538
	$\text{SIRLM}_{\text{SFT}}$	**0.429**	**0.599**	**0.379**	**0.531**	**0.556**	0.648	**0.549**	**0.668**	**0.599**	**0.749**	**0.388**	**0.550**
	$\text{SIRLM}_{\text{GRPO}}$	0.414	**0.595**	**0.383**	**0.542**	0.523	0.629	**0.541**	0.644	**0.593**	**0.744**	**0.381**	**0.544**
	Avg. gain (%)⋆	*+6.66*	*+5.90*	*+4.13*	*+5.77*	*+6.92*	*+5.11*	*+5.45*	*+6.01*	*+6.84*	*+6.64*	*+10.30*	*+12.05*

● RED-GNN can only obtain experimental results on 12 FullInd datasets partitioned from FB15k237, NELL995, and Wikidata68K.
† We reproduce ChatRule, MKGL, and KRLM on all datasets.
⋆ We calculate the average gain of the optimal value in the four training modes of SIRLM compared to all baselines.

GNN-based methods (NBFNet (Zhu et al., 2021), RED-GNN (Zhang & Yao, 2022), ULTRA (Galkin et al., 2024), and MOTIF (Huang et al., 2025a)), and **(4) LLM-based methods** (KICGPT (Wei et al., 2023), LSA (Li et al., 2025), ChatRule (Luo et al., 2025), MKGL (Guo et al., 2024), KG-FIT (Jiang et al., 2024a), FtG (Liu et al., 2025), and KRLM (Zhuo et al., 2026)).

Experimental settings. Based on previous work (Galkin et al., 2024), we adopt Mean Recurrent Rank (MRR) and top-10 Hit rate (Hit10) as evaluation metrics. In the main experiment, we use Qwen2.5-1.5b as the backbone of the SIRG module. We pre-train and fine-tune SIRLM using 4 A100 (40GB) GPUs. The more detailed settings of model hyperparameters are provided in **Appendix I**.

5.2 Main Results (RQ1)

Table 1 reports the overall performance of various methods across 36 KGR datasets under transductive, IndE, and FullInd settings. Overall, our proposed SIRLM achieves consistently strong and competitive results, outperforming prior methods on most datasets and metrics. Compared with embedding and rule-based approaches, LLM-based methods demonstrate clear advantages. This is because LLMs are better at understanding structural context and leveraging it for effective fact inference. GNN-based methods, supported by their ability to learn invariant graph representations, can capture general structural semantics across different KGs and thus achieve meaningful progress on inductive tasks.

In comparison to other baselines, LLM-based methods generally provide a holistic performance. However, most of them primarily focus on the transductive setting. Consequently, we reproduce open-source LLM-based methods (ChatRule, MKGL, and KRLM) for inductive reasoning. These methods are built upon GPT-4o mini or LLaMA2-7B and achieve competitive performance. In contrast, our SIRLM, despite being based on a 1.5B-scale LLM, surpasses these LLM-based approaches. This is primarily attributed to its effective internalization of structural knowledge and stricter adherence to logical semantics. More detailed experimental analysis can be found in Appendixes J.1.

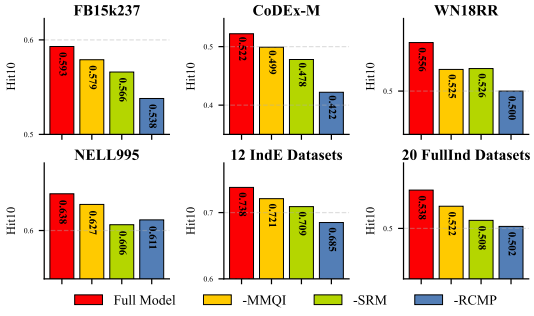


Figure 3: Hit10 of SIRLM_{PT} variants on different datasets, where the results in the IndE and FullInd scenarios are summarized as average values.

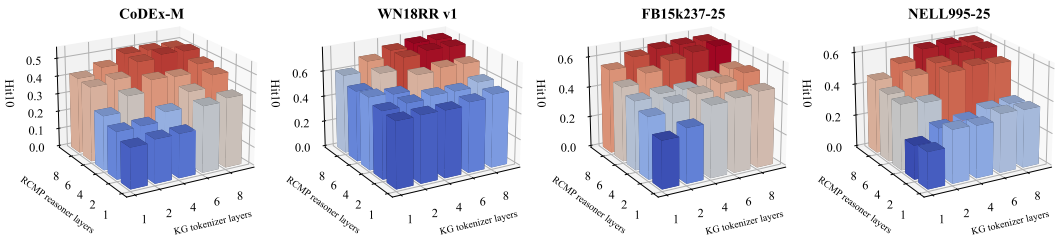


Figure 4: Hit10 of SIRLM$_{\text{E2E}}$ with different NBFNet layers N in KG tokenizer and RCMP reasoner.

5.3 Ablation Experiments (RQ2)

This section discusses the effectiveness of different modules in SIRLM. The experimental results are shown in Figure 3. Overall, the effectiveness of each ablation variant is inferior to that of the full model, especially in structural knowledge learning modules such as "SRM" and "RCMP". **Appendix J.2** provides detailed ablation variant settings and experimental results.

5.4 Parameter Analysis (RQ3)

Figures 4 and 5 illustrate the performance of SIRLM under different configurations of the NBFNet layers N, relation memory scale K, and the number of GRPO samples N. When the layers in both the KG tokenizer and the RCMP reasoner is set to $N = 6$, SIRLM achieves a good balance between reasoning accuracy and memory cost. K is related to the relation scale in the KG. Therefore, we uniformly set $K = 30$ in our experiments. When $M = 1$, the group relative advantage in Eq. (15) becomes ineffective, making it difficult for SIRLM to learn high-quality logical patterns, a phenomenon that is particularly pronounced in sparse KGs such as WN18RR v1 and CodeX-M. Therefore, we consistently set $M = 8$ in our experiments.

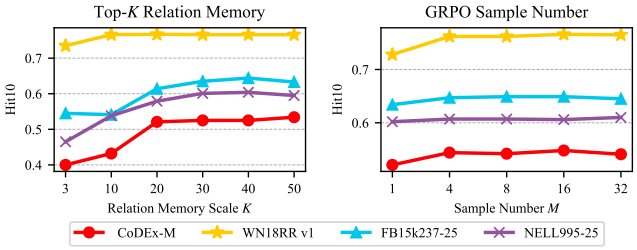


Figure 5: Hit10 of SIRLM$_{\text{E2E}}$ with different relation memory scale K and SIRLM$_{\text{GRPO}}$ with different GRPO sample number M.

5.5 Adaptability Analysis of LLM Backbones (RQ4)

This section evaluates the adaptability of SIRLM$_{\text{E2E}}$ across different LLM backbones. As shown in Table 2, SIRLM remains relatively stable across most backbones, with only a slight performance decline on Qwen2.5-0.5B. We attribute this gap to the limited capacity of smaller LLMs to assimilate structural knowledge. As shown in Figure 9 of **Appendix J.3**, Qwen2.5-0.5B exhibits the slowest convergence in rule-token generation accuracy, suggesting greater difficulty in aligning structural representations within its limited parameter space. As model capacity increases, SIRLM can more effectively internalize structural knowledge, leading to more stable reasoning performance.

Table 2: The performance of SIRLM$_{\text{E2E}}$ with different LLM backbones.

LLM Backbone	CoDEx-M		WN18RR v1		FB15k237-25		NELL995-25	
	MRR	Hit10	MRR	Hit10	MRR	Hit10	MRR	Hit10
Qwen2.5-0.5b	0.351	0.511	0.692	0.749	0.382	0.637	0.389	0.564
Qwen2.5-1.5b	0.368	0.525	0.705	0.766	0.386	0.635	0.407	0.601
Qwen2.5-7b	0.369	0.531	0.708	0.771	0.387	0.635	0.408	0.606
Llama-2-7b	0.362	0.528	0.711	0.781	0.387	0.632	0.405	0.602

6 Conclusion

This paper identifies a pervasive issue in LLM-based KGR, termed reasoning evidence perception drift, caused by the misalignment between KG structural representations and LLM parametric knowledge, which undermines both reasoning effectiveness and faithfulness. To address this issue, we propose the Structure-Internalized Rule Language Model (SIRLM), which integrates a structure-internalized rule generator with a KG tokenizer and a neuro-symbolic reasoner. SIRLM aligns parametric knowledge with KG structure through structural representation learning, rule generation, and faithfulness feedback, enabling evidence-grounded reasoning over KGs. Extensive experiments with 17 baselines on 36 KGR benchmarks show that SIRLM consistently outperforms existing methods across end-to-end training, pre-training, supervised fine-tuning, and post-training settings. **Appendix K** discusses its limitations and future directions.

AI USE STATEMENT

In this work, we have not used generative AI tools for any tasks requiring disclosure, and the remaining required disclosure tasks are not applicable to this work. We used generative AI tools only to edit the manuscript for grammar and readability. All AI-assisted edits were reviewed by the authors to ensure that they did not alter the technical content, scientific claims, or intended meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We confirm that our work has been conducted in accordance with the ICLR Code of Ethics (https://iclr.cc/public/CodeOfEthics). The study does not involve human subjects, sensitive personal data, or experiments that may cause harm to individuals or groups. The datasets used are publicly available and no personally identifiable information is included. Our methodology and findings are intended for academic purposes and do not pose foreseeable risks of misuse. We have carefully considered issues of fairness, bias, and privacy, and to the best of our knowledge, our research maintains integrity and complies with all applicable ethical standards.

REPRODUCIBILITY STATEMENT

We confirm that our study has reproducibility. Specifically, we have first submitted our desensitized project on anonymous GitHub (https://anonymous.4open.science/r/SIRLM-B35B/). The detailed pseudocode of the algorithm is provided in **Appendix F**. In addition, we provide specific details of the experimental conclusions in the main text, including dataset partitioning (**Appendix H**), hyperparameter settings (**Appendix I**), and ablation variant settings (**Appendix J.2**).

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Ivana Balazevic, Carl Allen, and Timothy Hospedales. TuckER: Tensor Factorization for Knowledge Graph Completion. In *EMNLP*, pp. 5185–5194. ACL, 2019.

Pablo Barceló, Mikhail Galkin, Christopher Morris, and Miguel A. Romero Orth. Weisfeiler and Leman Go Relational. In *LoG*, pp. 46. PMLR, 2022.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *NeurIPS*, pp. 2787–2795. Curran Associates, Inc., 2013.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal Neighbourhood Aggregation for Graph Nets. In *NeurIPS*, pp. 13260–13271. Curran Associates, Inc., 2020.

Yuanning Cui, Zequn Sun, and Wei Hu. A Prompt-Based Knowledge Graph Foundation Model for Universal In-Context Reasoning. In *NeurIPS*, pp. 7095–7124. Curran Associates, Inc., 2024.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. In *ICLR*. OpenReview.net, 2018.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D Knowledge Graph Embeddings. In *AAAI*, pp. 1811–1818. AAAI Press, 2018.

Mikhail Galkin, Etienne G. Denis, Jiapeng Wu, and William L. Hamilton. NodePiece: Compositional and Parameter-Efficient Representations of Large Knowledge Graphs. In *ICLR*. OpenReview.net, 2022.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards Foundation Models for Knowledge Graph Reasoning. In *ICLR*, pp. 31598–31619. OpenReview.net, 2024.

Genet Asefa Gesese, Harald Sack, and Mehwish Alam. RAILD: Towards Leveraging Relation Features for Inductive Link Prediction In Knowledge Graphs. In *IJCKG*, pp. 82–90. ACM, 2022.

Lingbing Guo, Zhongpu Bo, Zhuo Chen, Yichi Zhang, Jiaoyan Chen, Yarong Lan, Mengshu Sun, Zhiqiang Zhang, Yangyifei Luo, Qian Li, Qiang Zhang, Wen Zhang, and Huajun Chen. MKGL: Mastery of a Three-Word Language. In *NeurIPS*, volume 37, pp. 140509–140534. Curran Associates, Inc., 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*. OpenReview.net, 2022.

Manzong Huang, Chenyang Bu, Yi He, Xingrui Zhuo, and Xindong Wu. Relink: Constructing Query-Driven Evidence Graph On-the-Fly for GraphRAG. In *AAAI*, pp. 31202–31210. AAAI Press, 2026.

Xingyue Huang, Miguel Romero, İsmail İlkan Ceylan, and Pablo Barceló. A Theory of Link Prediction via Relational Weisfeiler-Leman on Knowledge Graphs. In *NeurIPS*, pp. 19714–19748. Curran Associates, Inc., 2023.

Xingyue Huang, Pablo Barceló, Michael M. Bronstein, İsmail İlkan Ceylan, Mikhail Galkin, Juan L. Reutter, and Miguel A. Romero Orth. How Expressive are Knowledge Graph Foundation Models? In *ICML*, pp. 25021–25058. PMLR, 2025a.

Xingyue Huang, Miguel A. Romero Orth, Pablo Barceló, Michael M. Bronstein, and İsmail İlkan Ceylan. Link Prediction with Relational Hypergraphs. *Trans. Mach. Learn. Res.*, 2025(1):1–42, 2025b.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE TNNLS*, 33(2):494–514, 2022.

Pengcheng Jiang, Lang Cao, Cao (Danica) Xiao, Parminder Bhatia, Jimeng Sun, and Jiawei Han. KG-FIT: Knowledge Graph Fine-Tuning Upon Open-World Knowledge. In *NeurIPS*, pp. 136220–136258. Curran Associates, Inc., 2024a.

Zhouyu Jiang, Ling Zhong, Mengshu Sun, Jun Xu, Rui Sun, Hui Cai, Shuhan Luo, and Zhiqiang Zhang. Efficient Knowledge Infusion via KG-LLM Alignment. In *Findings of ACL*, pp. 2986–2999. ACL, 2024b.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. KG-GPT: A General Framework for Reasoning on Knowledge Graphs Using Large Language Models. In *Findings of EMNLP*, pp. 9410–9421. ACL, 2023.

Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.*, 81(1):53–67, 2010.

Jaejun Lee, Chanyoung Chung, and Joyce Jiyoung Whang. InGram: Inductive Knowledge Graph Embedding via Relation Graphs. In *ICML*, volume 202, pp. 18796–18809. PMLR, 2023.

Miaomiao Li, Ke Liang, Yuping Lai, and Xinwang Liu. Knowledge Graph Reasoning Based on Information Enhancement and Subgraph Alignment. *IEEE TNNLS*, pp. 1–13, 2025.

Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, Fuchun Sun, and Kunlun He. A Survey of Knowledge Graph Reasoning on Graph Types: Static, Dynamic, and Multi-Modal. *IEEE TPAMI*, 46(12):9456–9478, 2024.

Ben Liu, Jihai Zhang, Fangquan Lin, Cheng Yang, and Min Peng. Filter-then-Generate: Large Language Models with Structure-Text Adapter for Knowledge Graph Completion. In *COLING*, pp. 11181–11195, 2025.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *ICLR*, pp. 14400–14423. OpenReview.net, 2024.

Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. ChatRule: Mining Logical Rules with Large Language Models for Knowledge Graph Reasoning. In *PAKDD*, pp. 314–325. Springer-Verlag, 2025.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A Comprehensive Overview of Large Language Models. *ACM TIST*, 16(5):106:1–106:72, 2025.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE TKDE*, 36(7):3580–3599, 2024.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs. In *ICLR*, 2021.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. DRUM: End-To-End Differentiable Rule Mining On Knowledge Graphs. In *NeurIPS*, pp. 15347–15357. Curran Associates, Inc., 2019.

Tara Safavi and Danai Koutra. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *EMNLP*, pp. 8328–8350. ACL, 2020.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*. OpenReview.net, 2019.

Komal K. Teru, Etienne G. Denis, and William L. Hamilton. Inductive Relation Prediction by Subgraph Reasoning. In *ICML*, volume 119, pp. 9448–9457. PMLR, 2020.

Kristina Toutanova and Danqi Chen. Observed Versus Latent Features for Knowledge Base and Text Inference. In *Workshop on CVSC*, pp. 57–66. ACL, 2015.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2071–2080. JMLR.org, 2016.

Guojia Wan and Bo Du. GaussianPath: A Bayesian Multi-Hop Reasoning Framework for Knowledge Graph Reasoning. In *AAAI*, pp. 4393–4401. AAAI Press, 2021.

Jiapu Wang, Kai Sun, Linhao Luo, Wei Wei, Yongli Hu, Alan Wee-Chung Liew, Shirui Pan, and Baocai Yin. Large language models-guided dynamic adaptation for temporal knowledge graph reasoning. In *NeurIPS*, pp. 8384–8410. Curran Associates, Inc., 2024a.

Kai Wang, Yuwei Xu, Zhiyong Wu, and Siqiang Luo. LLM as Prompter: Low-resource Inductive Reasoning on Arbitrary Knowledge Graphs. In *Findings of ACL*, pp. 3742–3759. ACL, 2024b.

Yanbin Wei, Qiushi Huang, Yu Zhang, and James T. Kwok. KICGPT: Large Language Model with Knowledge in Context for Knowledge Graph Completion. In *Findings of EMNLP*, pp. 8667–8683. ACL, 2023.

Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. From One Point to A Manifold: Orbit Models for Knowledge Graph Embedding. In *IJCAI*, pp. 1315–1321. ijcai.org, 2016.

Wenhan Xiong, Thien Hoang, and William Yang Wang. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *EMNLP*, pp. 564–573. ACL, 2017.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*. OpenReview.net, 2015.

Fan Yang, Zhilin Yang, and William W Cohen. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *NeurIPS*, pp. 2319–2328. Curran Associates, Inc., 2017.

Yichi Zhang, Zhuo Chen, Lingbing Guo, Yajing Xu, Wen Zhang, and Huajun Chen. Making Large Language Models Perform Better in Knowledge Graph Completion. In *ACM MM*, pp. 233–242. ACM, 2024a.

Yongqi Zhang and Quanming Yao. Knowledge Graph Reasoning with Relational Digraph. In *ACM WWW*, pp. 912–924. ACM, 2022.

Yucheng Zhang, Beatrice Bevilacqua, Mikhail Galkin, and Bruno Ribeiro. TRIX: A More Expressive Model for Zero-shot Domain Transfer in Knowledge Graphs. In *LoG Conference*. OpenReview.net, 2024b.

Jincheng Zhou, Beatrice Bevilacqua, and Bruno Ribeiro. A Multi-Task Perspective for Link Prediction with New Relation Types and Nodes. In *NeurIPS GLFrontiers Workshop*, 2023.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal A. C. Xhonneux, and Jian Tang. Neural Bellman-Ford Networks: A General Graph Neural Network Framework for Link Prediction. In *NeurIPS*, pp. 29476–29490. Curran Associates, Inc., 2021.

Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. Neural-Symbolic Models for Logical Queries on Knowledge Graphs. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27454–27478. PMLR, 2022.

Xingrui Zhuo, Shirui Pan, Jiapu Wang, Gongqing Wu, Zan Zhang, Rui Li, Zizhong Wei, and Xindong Wu. Progressive Prefix-Memory Tuning for Complex Logical Query Answering on Knowledge Graphs. In *IJCAI*, pp. 3716–3724. ijcai.org, 2025a.

Xingrui Zhuo, Jiapu Wang, Gongqing Wu, Shirui Pan, and Xindong Wu. Effective instruction parsing plugin for complex logical query answering on knowledge graphs. In *ACM WWW*, pp. 4780–4792. ACM, 2025b.

Xingrui Zhuo, Jiapu Wang, Gongqing Wu, Zhongyuan Wang, Jichen Zhang, Shirui Pan, and Xindong Wu. Knowledge Reasoning Language Model: Unifying Knowledge and Language for Inductive Knowledge Graph Reasoning. In *ICLR*, pp. 122329–122356, 2026.

A Design Details of Query Instructions

Given a query triplet $(e_h, r_q, ?)$, we first provide its schema of a query instruction below:

Schema of the Query Instruction for Rule Generation

Suppose you are a linguistic expert who is learning a new rule language. Given the following head entity and query relation:

Head entity: <Ent e_h>

Query relations: <Rel r_q>

Generate the following rule language: <Rel r_q>=$\underbrace{< \text{Rel } r_1 >< \text{Rel } r_3 >< \text{Rel } r_2 >< \text{END} >}_{\text{mask for generation}}$

Our query instruction consists of a fixed text prompt along with structural representation placeholders that vary depending on the query triplet. Specifically, $<$ Ent e_h $>$ and $<$ Rel r_q $>$ correspond to the structural representations of e_h and r_q, respectively. The masked portion includes the structural representations of atomic relations in the rule body, $<$ Rel r_1 $><$ Rel r_3 $><$ Rel r_2 $>$, as well as the rule termination token $<$ END $>$, all of which are derived from TKN_{KG} in Eq. (6).

This instruction format integrates textual tokes with the structural representations of a KG, enabling the LLM to further learn the contextual semantics of entities and relations within the structural space. Moreover, since the structural representations provided by NBFNet inherently encode high-order graph information, we can deliver richer structural information to the LLM using fewer tokens. In addition, this instruction format does not rely on explicit textual names of entities and relations, making it suitable for anonymized KGR scenarios.

We compare the instruction lengths designed by several recent LLM-based KGR methods in Table 3, demonstrating that our constructed instructions achieve more efficient utilization of computational resources.

Table 3: Average instruction length of MKG, KRLM, and SIRLM in 36 KGR datasets.

Model	MKGL (Guo et al., 2024)	KRLM (Zhuo et al., 2026)	SIRLM
Avg. Length	115.66±3.65	120.50±4.12	50.73±0.67

B Relational Graph Construction

Unlike a typical KG, a relational graph (Galkin et al., 2024; Huang et al., 2025a) is used to describe the relative states between relations. As shown in Figure 6, in our study, the motifs connecting relation nodes in a relational graph are essentially a set of relation-oriented hyperedges that is defined as $\mathcal{R}^*$ in Eq. (1).

In practical implementation, we use sparse matrices to construct the adjacency matrix of the relational graph. Giving a KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ defined as a graph with multiple directed edges, and its adjacency matrix can be represented as $\boldsymbol{A} \in \mathbb{R}^{\mathcal{E}\times\mathcal{R}\times\mathcal{E}}$. Subsequently, we construct the adjacency matrix by performing the maximum scatter operation on the head and tail node dimensions, resulting in two sparse matrices $\boldsymbol{A}_h \in \mathbb{R}^{\mathcal{E}\times\mathcal{R}}$ and $\boldsymbol{A}_t \in \mathbb{R}^{\mathcal{R}\times\mathcal{E}}$. $\boldsymbol{A}_h$ represents the out-degree edge of any relation from any node and $\boldsymbol{A}_t$ represents the in-degree edge of any relation pointing to any node. Then, the eight types of motif edges shown in Figure 6 can be obtained through a sparse matrix multiplication

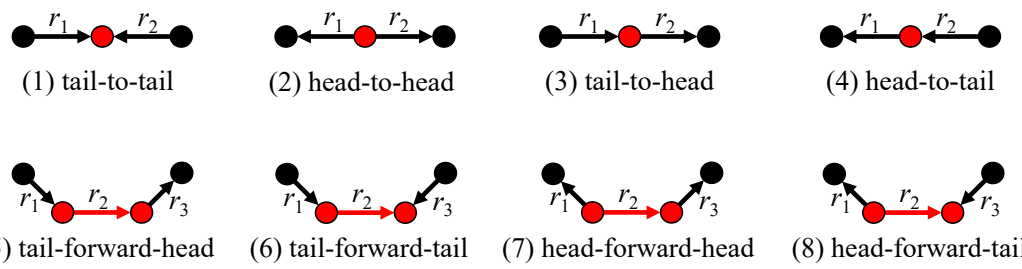


Figure 6: Motif edges of relations in a KG, whcih can be grouped into three binary edges and four ternary edges. A binary edge represents the state of the entity shared by two ordered relations, including **(1) tail-to-tail**: a shared tail entity, **(2) head-to-head**: a shared head entity, **(3) tail-to-head**: the tail entity of the previous relation being the head entity of the next relation, and **(4) head-to-tail**: the head entity of the previous relation being the tail entity of the next relation. A ternary edge represents the state of a forward triplet shared by two ordered relations, which is a hyperedge containing three relation nodes, including **(5) tail-forward-head**: the forward triplet as the tail of the previous relation and the head of the next relation, **(6) tail-forward-tail**: the forward triplet as the shared tail of two relations, **(7) head-forward-head**: the forward triplet as the shared head of two relations, and **(8) head-forward-tail**: the forward triplet as the head of the previous relation and the tail of the next relation.

(spmm) operator:

$$\text{Binary edges} = \begin{cases} \textbf{Tail-to-tail}\ (t2t)\text{: } \boldsymbol{A}_{t2t} = \text{spmm}(\boldsymbol{A}_t, \boldsymbol{A}_t^{\mathrm{T}}) \in \mathbb{R}^{\mathcal{R}\times\mathcal{R}} \\ \textbf{Head-to-head}\ (h2h)\text{: } \boldsymbol{A}_{h2h} = \text{spmm}(\boldsymbol{A}_h^{\mathrm{T}}, \boldsymbol{A}_h) \in \mathbb{R}^{\mathcal{R}\times\mathcal{R}} \\ \textbf{Tail-to-head}\ (t2h)\text{: } \boldsymbol{A}_{t2h} = \text{spmm}(\boldsymbol{A}_t, \boldsymbol{A}_h) \in \mathbb{R}^{\mathcal{R}\times\mathcal{R}} \\ \textbf{Head-to-tail}\ (h2t)\text{: } \boldsymbol{A}_{h2t} = \text{spmm}(\boldsymbol{A}_h^{\mathrm{T}}, \boldsymbol{A}_t^{\mathrm{T}}) \in \mathbb{R}^{\mathcal{R}\times\mathcal{R}} \end{cases}$$

$$\text{Ternary edges} = \begin{cases} \textbf{Tail-forward-head}\ (tfh)\text{: } \boldsymbol{A}_{tfh} = \text{spmm}(\boldsymbol{A}_t, \boldsymbol{A}, \boldsymbol{A}_h) \in \mathbb{R}^{\mathcal{R}\times\mathcal{R}\times\mathcal{R}} \\ \textbf{Tail-forward-tail}\ (tft)\text{: } \boldsymbol{A}_{tft} = \text{spmm}(\boldsymbol{A}_t, \boldsymbol{A}, \boldsymbol{A}_t^{\mathrm{T}}) \in \mathbb{R}^{\mathcal{R}\times\mathcal{R}\times\mathcal{R}} \\ \textbf{Head-forward-heaed}\ (hfh)\text{: } \boldsymbol{A}_{tft} = \text{spmm}(\boldsymbol{A}_h^{\mathrm{T}}, \boldsymbol{A}, \boldsymbol{A}_h) \in \mathbb{R}^{\mathcal{R}\times\mathcal{R}\times\mathcal{R}} \\ \textbf{Head-forward-tail}\ (hft)\text{: } \boldsymbol{A}_{hft} = \text{spmm}(\boldsymbol{A}_h^{\mathrm{T}}, \boldsymbol{A}, \boldsymbol{A}_t^{\mathrm{T}}) \in \mathbb{R}^{\mathcal{R}\times\mathcal{R}\times\mathcal{R}} \end{cases}$$

C Construction Details of the KG tokenizer and the RCMP Reasoner

Given that relational graphs contain ternary edges, there are certain differences in the construction details of NBFNet between the original KGs and the relational graphs in the KG tokenizer and the RCMP Reasoner.

For NBFNet on a relational graph, we need to consider the positional information of the relational nodes in the hyperedges. Therefore, Eq. (1) can be concretized as

$$\begin{aligned} \boldsymbol{r}_{j|q}^{(0)} &= \mathbb{I}(r_j = r_q) * \mathbf{1}^d, \\ \boldsymbol{r}_{j|q}^{(n)} &= \sigma\Big(\boldsymbol{W}_r^{(n-1)}\Big[\boldsymbol{r}_{j|q}^{(n-1)} || \sum_{r^*\in\mathcal{R}^*} \boldsymbol{r}^* \odot \big(\bigodot_{r_z\in\mathcal{N}_{r^*}(r_j)} (\boldsymbol{r}_{z|q}^{(n-1)} + \boldsymbol{p}_z)\big)\Big]\Big), \boldsymbol{r}^* \in \boldsymbol{R}^*, \end{aligned} \tag{17}$$

where $\sigma(\cdot)$ is a ReLU activation function, $\odot$ is the Hadamard product operator, and $\boldsymbol{W}_r^{(n-1)} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{p}_z \in \mathbb{R}^d$ is a trainable parameter matrix and a position embedding, respectively.

For the original KG, NBFNet only processes directed binary edges, so position information can be ignored. Therefore, Eq. (2) can be concretized as

$$\begin{aligned} \boldsymbol{e}_{i|h}^{(0)} &= \mathbb{I}(e_i = e_h) * \boldsymbol{r}_{q|q}^{(N)}, \\ \boldsymbol{e}_{i|h}^{(n)} &= \sigma\Big(\boldsymbol{W}_e^{(n-1)}\Big[\boldsymbol{e}_{i|h}^{(n-1)} || \sum_{r_j\in\mathcal{R}} \sum_{e_z\in\mathcal{N}_{r_j}(e_i)} \boldsymbol{r}_{j|q}^{(N)} \odot \boldsymbol{e}_{z|h}^{(n-1)}\Big]\Big), \end{aligned} \tag{18}$$

where $\boldsymbol{W}_e^{(n-1)} \in \mathbb{R}^{d\times d}$ is a trainable parameter matrix.

Similarly, Eqs. (11) and (12) can be concretized as Eqs. (19) and (20), respectively:

$$\begin{aligned}\hat{\boldsymbol{r}}_{j|q} &= \sum_{<r,\boldsymbol{v}>\in U} \mathbb{I}(r_j = r) * \boldsymbol{v}, \text{ where } U = \{< r_q, \mathbf{1}^d >\} \cup \{< r_z, f_{\text{down}}(\boldsymbol{h}_{L+z}) >\}_{z=1}^{\varepsilon} \\ \hat{\boldsymbol{r}}_{j|q}^{(n)} &= \sigma\Big(\hat{\boldsymbol{W}}_r^{(n-1)}\Big[\hat{\boldsymbol{r}}_{j|q}^{(n-1)} || \sum_{r^*\in\mathcal{R}^*} \hat{\boldsymbol{r}}^* \odot \big(\bigodot_{r_z\in\mathcal{N}_{r^*}(r_j)} (\hat{\boldsymbol{r}}_{z|q}^{(n-1)} + \boldsymbol{p}_z)\big)\Big]\Big), \hat{\boldsymbol{r}}^* \in \hat{\boldsymbol{R}}^*,\end{aligned} \tag{19}$$

$$\begin{aligned}\hat{\boldsymbol{e}}_{i|h}^{(0)} &= \mathbb{I}(e_i = e_h) * \hat{\boldsymbol{r}}_{q|q}^{(N)}, \\ \hat{\boldsymbol{e}}_{i|h}^{(n)} &= \sigma\Big(\hat{\boldsymbol{W}}_e^{(n-1)}\Big[\hat{\boldsymbol{e}}_{i|h}^{(n-1)} || \sum_{r_j\in\mathcal{R}} \sum_{e_z\in\mathcal{N}_{r_j}(e_i)} \hat{\boldsymbol{r}}_{j|q}^{(N)} \odot \hat{\boldsymbol{e}}_{z|h}^{(n-1)}\Big]\Big).\end{aligned} \tag{20}$$

Here, $\hat{\boldsymbol{W}}_r^{(n-1)}, \hat{\boldsymbol{W}}_e^{(n-1)} \in \mathbb{R}^{d\times d}$ are two trainable parameter matrix.

D Discussion of the SRM In-context Layer

This section discusses the effectiveness of the proposed SRM mechanism from the perspectives of structural representation alignment and semantic drift suppression. First, we provide the following explanation of the assumption and definitions required for subsequent analysis.

Definition 1 (**In-context learning for the last token**). *Given a query triplet $< e_h, r_q, ? >$ and the corresponding instruction representation $\boldsymbol{X} \in \mathbb{R}^{L\times F}$, according to Eq. (4), the hidden state of the last token $\boldsymbol{x}_L \in \boldsymbol{X}$ in the vanilla in-context learning module can be expressed as:*

$$\boldsymbol{h}_L^{\text{base}} = \sum_{i\le L} \alpha_i f_V(\boldsymbol{x}_i),$$

$$\alpha_i = \frac{\exp(a_i)}{\sum_{j\le L}\exp(a_j)}, \quad a_i = \frac{f_Q(\boldsymbol{x}_L)[f_K(\boldsymbol{x}_i)]^T}{\sqrt{F}}.$$

Then, we introduce the structural relation memory $\boldsymbol{R}_{\text{mem}} = \{\boldsymbol{r}_k \in \boldsymbol{R}_{|q}\}_{k=1}^K$ from Eq. (7) into the in-context learning module, the hidden state of the last token in Eq. (8) can be expressed as:

$$\boldsymbol{h}_L^{\text{SRM}} = \sum_{i\le L} \hat{\alpha}_i f_V(\boldsymbol{x}_i) + \sum_{k\le K} \beta_k m_V(\boldsymbol{r}_k),$$

$$\hat{\alpha}_i = \frac{\exp(a_i)}{\sum_{j\le L}\exp(a_j) + \sum_{z\le K}\exp(b_z)}, \quad \beta_k = \frac{\exp(b_k)}{\sum_{j\le L}\exp(a_j) + \sum_{z\le K}\exp(b_z)}, \quad b_k = \frac{f_Q(\boldsymbol{x}_L)[m_K(\boldsymbol{r}_k)]^T}{\sqrt{F}}.$$

Definition 2 (**Structural subspace and structural observables**). *Given a query triplet $< e_h, r_q, ? >$, define the structural subspace constructed by SIRLM as:*

$$\mathcal{S} = \text{span}(\{f_{\text{up}}(\boldsymbol{r}) | \boldsymbol{r} \in \boldsymbol{R}_{|q}\} \cup \{m_V(\boldsymbol{r}_k) | \boldsymbol{r}_k \in \boldsymbol{R}_{\text{mem}}\}_{k=1}^K).$$

Let $\boldsymbol{\Pi}_{\mathcal{S}}$ denote the orthogonal projection operator onto $\mathcal{S}$. Then any $\boldsymbol{x} \in \mathbb{R}^F$ can be uniquely decomposed in $\mathcal{S}$ as:

$$\boldsymbol{x} = \boldsymbol{\Pi}_{\mathcal{S}}\boldsymbol{x} + \boldsymbol{\Pi}_{\mathcal{S}}^{\perp}\boldsymbol{x},$$

where $\boldsymbol{\Pi}_{\mathcal{S}}\boldsymbol{x} \in \mathcal{S}$ represents the structural component of $\boldsymbol{x}$ and $\boldsymbol{\Pi}_{\mathcal{S}}^{\perp}\boldsymbol{x}$ is the semantic drift component orthogonal to $\mathcal{S}$.

Furthermore, define:

$$\boldsymbol{u}_q = \frac{\boldsymbol{\Pi}_{\mathcal{S}} f_{\text{up}}(\boldsymbol{r}_{q|q})}{\|\boldsymbol{\Pi}_{\mathcal{S}} f_{\text{up}}(\boldsymbol{r}_{q|q})\|_2}, \quad \boldsymbol{r}_{q|q} \in \boldsymbol{R}_{|q},$$

which represents extracting a unit direction from the structural representation of r_q within the structural subspace $\mathcal{S}$, serving as the structural anchor of r_q.

Based on $\boldsymbol{u}_q$ and $\boldsymbol{\Pi}_{\mathcal{S}}$, two structural observables can be defined:

- ***Structural alignment score**: $A_q(\boldsymbol{x}) = \boldsymbol{u}_q^{\mathrm{T}}\boldsymbol{x}$, which measures the relevance of any $\boldsymbol{x}$ to $\boldsymbol{r}_q$ within the structural subspace $\mathcal{S}$.*

- ***Semantic drift magnitude***: *$D_{\mathcal{S}}(\boldsymbol{x}) = \boldsymbol{u}_q^{\mathrm{T}}\boldsymbol{x}$, which measures the relevance of any $\boldsymbol{x}$ to $\boldsymbol{r}_q$ within the structural subspace $\mathcal{S} = \|\boldsymbol{\Pi}_{\mathcal{S}}^{\perp}\boldsymbol{x}\|_2$, which quantifies the component of any $\boldsymbol{x}$ orthogonal to the structural subspace $\mathcal{S}$.*

Assumption A. Given a strictly increasing function $\phi(\cdot)$ and a strictly decreasing function $\psi(\cdot)$, for all Top-K relations selected in Eq. (7), we have:

$$A_q(m_V(\boldsymbol{r}_k)) = \phi(s_{\mathrm{rel}}^{(k)}), \quad D_{\mathcal{S}}(m_V(\boldsymbol{r}_k)) = \psi(s_{\mathrm{rel}}^{(k)}).$$

Assume there exists a threshold s_0 such that $\{s_{\mathrm{rel}}^{(k)} > s_0\}_{k=1}^{K}$. Then for $\boldsymbol{h}_L^{\mathrm{base}}$, we have:

$$A_q(\boldsymbol{h}_L^{\mathrm{base}}) \le \phi(s_0), \quad D_{\mathcal{S}}(\boldsymbol{h}_L^{\mathrm{base}}) \ge \psi(s_0).$$

That is, the aggregate result of the vanilla in-context learning module has a clear separation boundary from the structural reasoning subspace of the query triplet.

Based on the above definitions and assumptions, we present the following proposition.

Proposition 1. *Given a query triplet $< e_h, r_q, ? >$ and its corresponding structural subspace $\mathcal{S}$, let $\boldsymbol{h}_L^{\mathrm{SRM}}$ and $\boldsymbol{h}_L^{\mathrm{base}}$ be the outputs obtained from the SRM-based and vanilla in-context learning modules, respectively. Then, the former exhibits stronger structural representation alignment and better suppression of semantic drift than the latter, i.e.,*

$$A_q(\boldsymbol{h}_L^{\mathrm{SRM}}) > A_q(\boldsymbol{h}_L^{\mathrm{base}}), \quad D_{\mathcal{S}}(\boldsymbol{h}_L^{\mathrm{SRM}}) < D_{\mathcal{S}}(\boldsymbol{h}_L^{\mathrm{base}}).$$

Proof. We prove the two claims separately.

(I) Structural alignment improvement. By Definitions 1 and 2, we have

$$\begin{cases} A_q(\boldsymbol{h}_L^{\mathrm{base}}) = \frac{\sum_{i\le L}\exp(a_i)A_q\left(f_V(\boldsymbol{x}_i)\right)}{\sum_{j\le L}\exp(a_j)} \\ A_q(\boldsymbol{h}_L^{\mathrm{SRM}}) = \frac{\sum_{i\le L}\exp(a_i)A_q\left(f_V(\boldsymbol{x}_i)\right)}{\sum_{j\le L}\exp(a_j)+\sum_{z\le K}\exp(b_z)} + \sum_{k\le K}\beta_k A_q\left(m_V(\boldsymbol{r}_k)\right) \end{cases}$$

$$\Rightarrow A_q(\boldsymbol{h}_L^{\mathrm{SRM}}) = \frac{\sum_{j\le L}\exp(a_j)}{\sum_{j\le L}\exp(a_j)+\sum_{z\le K}\exp(b_z)} A_q(\boldsymbol{h}_L^{\mathrm{base}}) + \sum_{k\le K}\beta_k A_q\left(m_V(\boldsymbol{r}_k)\right).$$

Let $c = \frac{\sum_{j\le L}\exp(a_j)}{\sum_{j\le L}\exp(a_j)+\sum_{z\le K}\exp(b_z)}$, then $\sum_{k\le K}\beta_k = 1 - c$. Further, define $\overline{A}_{\mathrm{mem}} = \frac{\sum_{k\le K}\beta_k A_q\left(m_V(\boldsymbol{r}_k)\right)}{\sum_{k\le K}\beta_k}$, $A_q(\boldsymbol{h}_L^{\mathrm{SRM}})$ can be rewritten as:

$$A_q(\boldsymbol{h}_L^{\mathrm{SRM}}) = A_q(\boldsymbol{h}_L^{\mathrm{base}}) + (1-c)(\overline{A}_{\mathrm{mem}} - A_q(\boldsymbol{h}_L^{\mathrm{base}})).$$

Therefore, it suffices to prove that $\overline{A}_{\mathrm{mem}} > A_q(\boldsymbol{h}_L^{\mathrm{base}})$.

According to **Assumption A** and the definition of $\overline{A}_{\mathrm{mem}}$, we have:

$$\begin{cases} A_q(m_V(\boldsymbol{r}_k)) = \phi(s_{\mathrm{rel}}^{(k)}) > \phi(s_0) \\ A_q(\boldsymbol{h}_L^{\mathrm{base}}) \le \phi(s_0) \end{cases} \Rightarrow \overline{A}_{\mathrm{mem}} > A_q(\boldsymbol{h}_L^{\mathrm{base}}).$$

Therefore, $A_q(\boldsymbol{h}_L^{\mathrm{SRM}}) > A_q(\boldsymbol{h}_L^{\mathrm{base}})$ is proven.

(II) Suppression of semantic drift. Following a similar procedure as in the previous proof, and based on Definitions 1 and 2, we obtain:

$$\begin{cases} D_{\mathcal{S}}(\boldsymbol{h}_L^{\mathrm{base}}) = \left\| \frac{\sum_{i\le L}\exp(a_i)\boldsymbol{\Pi}_{\mathcal{S}}^{\perp} f_V(\boldsymbol{x}_i)}{\sum_{j\le L}\exp(a_j)} \right\|_2 \\ D_{\mathcal{S}}(\boldsymbol{h}_L^{\mathrm{SRM}}) = \left\| \frac{\sum_{i\le L}\exp(a_i)\boldsymbol{\Pi}_{\mathcal{S}}^{\perp} f_V(\boldsymbol{x}_i)}{\sum_{j\le L}\exp(a_j)+\sum_{z\le K}\exp(b_z)} + \sum_{k\le K}\beta_k \boldsymbol{\Pi}_{\mathcal{S}}^{\perp} m_V(\boldsymbol{r}_k) \right\|_2 \end{cases}$$

$$\Rightarrow D_{\mathcal{S}}(\boldsymbol{h}_L^{\mathrm{SRM}}) \le c D_{\mathcal{S}}(\boldsymbol{h}_L^{\mathrm{base}}) + \sum_{k\le K}\beta_k D_{\mathcal{S}}\left(m_V(\boldsymbol{r}_k)\right).$$

Let $\overline{D}_{\text{mem}} = \frac{\sum_{k \le K} \beta_k D_{\mathcal{S}}\left(m_V(\boldsymbol{r}_k)\right)}{\sum_{k \le K} \beta_k}$, then $D_{\mathcal{S}}(\boldsymbol{h}_L^{\text{SRM}})$ admits the following upper bound:

$$D_{\mathcal{S}}(\boldsymbol{h}_L^{\text{SRM}}) \le D_{\mathcal{S}}(\boldsymbol{h}_L^{\text{base}}) + (1-c)(\overline{D}_{\text{mem}} - D_{\mathcal{S}}(\boldsymbol{h}_L^{\text{base}})).$$

Thus, it suffices to prove that $\overline{D}_{\text{mem}} < D_{\mathcal{S}}(\boldsymbol{h}_L^{\text{base}})$.

According to **Assumption A** and the definition of $\overline{D}_{\text{mem}}$, we have:

$$\begin{cases} D_{\mathcal{S}}(m_V(\boldsymbol{r}_k)) = \psi(s_{\text{rel}}^{(k)}) < \psi(s_0) \\ D_{\mathcal{S}}(\boldsymbol{h}_L^{\text{base}}) \ge \psi(s_0) \end{cases} \Rightarrow \overline{D}_{\text{mem}} < D_{\mathcal{S}}(\boldsymbol{h}_L^{\text{base}}).$$

Therefore, $D_{\mathcal{S}}(\boldsymbol{h}_L^{\text{SRM}}) < D_{\mathcal{S}}(\boldsymbol{h}_L^{\text{base}})$ is proven. □

Discussion. The proof above establishes the effectiveness of Eq. (8). The SRM branch does not merely append extra structural representation, it actively reorganizes the hidden state by aligning its structural component and suppressing text-only semantic bias. Because the next-relation predictor in Eqs. (9) and (10) operates directly on this hidden state, the hidden-state improvement transfers into a larger score margin for the correct structural relation, thereby promoting the generation of KG-grounded rules.

E Discussion of the RCMP Mechanism

This section discusses the effectiveness of the proposed RCMP mechanism. Based on the theoretical foundation of the previous studies (Huang et al., 2023; 2025b), we start with the relational Weisfeiler-Leman (WL) test (Barceló et al., 2022) to analyze the effectiveness of RCMP in terms of structural invariance learning, lower bound of model expression, and stricter reasoning scenarios.

E.1 A Two-stage WL Test for the SIL Mechanism

Before formally discussing the effectiveness of RCMP, we first provide a theoretical background for the WL test of the primary SIL mechanism (Huang et al., 2025a) without RCMP. Giving a KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ and a corresponding relational graph $\mathcal{G}_r = (\mathcal{R}, \mathcal{R}^*, \mathcal{T}^*)$, the WL test of the SIL mechanism can be defined as a *two-stage coloring scheme*.

Stage 1: relation coloring on $\mathcal{G}_r$. According to Eq. (1), giving a query relation $r_q \in \mathcal{R}$, the color of relation $r_j \in \mathcal{R}$ at iteration n is denoted by $\text{col}_r^{(n)}(r_j|r_q)$ and initialized by

$$\text{col}_r^{(0)}(r_j|r_q) = \mathbb{I}(r_j = r_q) \cdot \mathbf{1}.$$

It is then updated as

$$\text{col}_r^{(n+1)}(r_j|r_q) = \text{HASH}\Big(\text{col}_r^{(n)}(r_j|r_q), \{\!\{(\{(\text{col}_r^{(n)}(r_s|r_q), p_s) \mid (r_s, p_s) \in \mathcal{N}_{r^*}(r_j)\}, r^*) \mid r^* \in \mathcal{R}^*\}\!\}\Big),$$

where $\mathcal{N}_{r^*}(r_j)$ represents the set of all neighbors of r_j on r^*. Since r^* is a hyperedge, $\mathcal{N}_{r^*}(r_j)$ stores each neighbor r_s and its position p_s on r^*. $\text{HASH}(\cdot)$ is an abstract injective coloring function (Barceló et al., 2022) that compresses old colors and neighborhood information into a new color label. As long as the input is different, the output color will be different.

Stage 2: entity coloring on $\mathcal{G}$. Given a fixed number N of first-stage iterations, the color of an entity e_v in a candidate triplet $< e_u, r_q, e_v >$ at iteration ℓ is denoted by $\text{col}_e^{(\ell)}(e_v|e_u, r_q)$. It is initialized by

$$\text{col}_e^{(0)}(e_v|e_u, r_q) = \mathbb{I}(e_v = e_u) \cdot \text{col}_r^{(N)}(r_q|r_q),$$

and updated as

$$\text{col}_e^{(\ell+1)}(e_v|e_u, r_q) = \text{HASH}\Big(\text{col}_e^{(\ell)}(e_v|e_u, r_q), \{\!\{(\text{col}_e^{(\ell)}(e_w|e_u, r_q), \text{col}_r^{(N)}(r_j|r_q)) \mid e_w \in \mathcal{N}_{r_j}(e_v),\ r_j \in \mathcal{R}\}\!\}\Big).$$

This two-stage WL test characterizes the separating power of the SIL mechanism (Huang et al., 2025a): it is both an upper bound and a tight characterization under injective message-passing choices.

E.2 A TWO-STAGE WL TEST FOR THE RCMP MECHANISM

This section provides a detailed elaboration on the effectiveness of the proposed RCMP based on **Appendix E.1**. First, we outline the overall pipeline of the proposed SIRLM as **Definition 3** for subsequent analysis.

Definition 3. *Giving a KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ and a corresponding relational graph $\mathcal{G}_r = (\mathcal{R}, \mathcal{R}^*, \mathcal{T}^*)$, SIRLM processes a query $< e_u, r_q, ? >$ through the following stages:*

Stage 1: structural tokenization. *We first run Eqs. (1) and (2) to obtain structural token embeddigns of relation $\boldsymbol{r}_{j|q} \in \boldsymbol{R}_{|q}$ and entity $\boldsymbol{e}_{i|u} \in \boldsymbol{E}_{|u}$. **These embeddings are used as structure-only tokens; no entity names, relation names, or textual attributes are used**. In particular, only $\boldsymbol{e}_{u|u}$ and $\boldsymbol{r}_{q|q}$ are exposed to the rule generator as query context.*

Stage 2: structure-aware rule generation. *Let $\mathcal{P}_{\text{sys}}$ be a fixed system prompt. For a query $r_q(e_u, ?)$, define the input prompt as*

$$\Pi(e_u, r_q) = [\mathcal{P}_{\text{sys}};\ \boldsymbol{e}_{u|u};\ \boldsymbol{r}_{q|q}\,].$$

LLM generates a rule consisting of a sequence of atomic relation

$$\rho(e_u, r_q) = r_{1|\Pi(e_u,r_q)}, \ldots, r_{Z|\Pi(e_u,r_q)} \in \mathcal{R}$$

and corresponding hidden states

$$\boldsymbol{h}_{1|\Pi(e_u,r_q)}, \ldots, \boldsymbol{h}_{Z|\Pi(e_u,r_q)}.$$

At z-th generation iteration, the last hidden state $\boldsymbol{h}_{z-1|\Pi(e_u,r_q)}$ is computed by a LLM fed with $\Pi(e_u, r_q)$ and the previously $z-1$ generated atomic relations. The next atomic relation is then selected by a lookup over the structured relation-token bank $\boldsymbol{R}_{|q}$:

$$r_{z|\Pi(e_u,r_q)} = \arg\max_{r_j \in R} \text{Score}\big(\boldsymbol{h}_{z-1|\Pi(e_u,r_q)}, \boldsymbol{r}_{j|q}\big),$$

where $\text{Score}(\cdot)$ is a relation-token lookup scorer.

Stage 3: RCMP initialization of $\mathcal{G}_r$. *For each relation $r \in \mathcal{R}$, define the accumulated rule representation as*

$$\boldsymbol{x}(r|e_u, r_q) = \sum_{z\,:\, r_{z|\Pi(e_u,r_q)}=r} \boldsymbol{h}_{z|\Pi(e_u,r_q)},$$

with the convention that $\boldsymbol{x}(r|e_u, r_q) = \mathbf{0}$ if r does not occur in $\rho(e_u, r_q)$. This additive definition directly handles repeated relations, that is, every occurrence contributes its own autoregressive hidden state, and all such occurrence-specific hidden states are accumulated.

Giving the generated rule $\rho(e_u, r_q)$ and the accumulated rule representation $\boldsymbol{x}(\cdot|e_u, r_q)$, the RCMP initialization of $\mathcal{G}_r$ can be concretized as

$$\widetilde{\text{INIT}}_1(r_j|e_u, r_q) = \mathbb{I}(r_j = r_q) \cdot \mathbf{1} + \boldsymbol{x}(r_j|e_u, r_q),$$

where $\mathbf{1} \neq \mathbf{0}$ are fixed vectors. Equivalently, if the query relation r_q appears in the generated rule body, then its node in the relational graph is initialized by the anchor vector $\mathbf{1}$ plus the sum of all hidden states associated with occurrences of r_q in $\rho(e_u, r_q)$. Likewise, if any non-query relation r_j appears multiple times in $\rho(e_u, r_q)$, then its initialization is the sum of the hidden states associated with all of its occurrences.

It should be emphasized that RCMP only changes the initialization strategy of the relational graph $\mathcal{G}_r$ in SIL with $\widetilde{\text{INIT}}_1(r_j|e_u, r_q)$, while keeping $\mathcal{G}_r$ and all relation and entity message-passing operators unchanged.

Assumption B. We make the following assumptions.

1. **Structure-token invariance**. The base embeddings $\boldsymbol{r}_{j|q} \in \boldsymbol{R}_{|q}$ and $\boldsymbol{e}_{i|u} \in \boldsymbol{E}_{|u}$ are structurally invariants in **Definition 3**.

2. **Equivariance of LLM generation for structural relation tokens**. For every isomorphism $h = (\pi, \phi) : \mathcal{G} \to \mathcal{G}'$, the autoregressive decoder and the lookup rule are equivariant with respect to the induced renaming of the structural tokens, *i.e.*,

$$\boldsymbol{h}_{z|\Pi(e_u,r_q)} = \boldsymbol{h}_{z|\Pi(\pi(e_u),\phi(r_q))}, \qquad r_{z|\Pi(\pi(e_u),\phi(r_q))} = \phi(r_{z|\Pi(e_u,r_q)}).$$

3. **Anchor separation of relations**. Query relations are disjoint from non-query relations, *i.e.*,
$$\mathbf{1} + \boldsymbol{x}(r_j|e_u, r_q) \neq \boldsymbol{x}(r'_j|e'_u, r'_q) \qquad \text{whenever } r'_j \neq r'_q.$$
Equivalently, $\widetilde{\mathrm{INIT}}_1(r_j|e_u, r_q) = \widetilde{\mathrm{INIT}}_1(r'_j|e'_u, r'_q)$ implies $r_j = r_q$ if and only if $r'_j = r'_q$.

4. **Injective operator in RCMP**. The update, aggregation, and message functions in both the relation encoder and the entity encoder are injective, exactly as in the constructive direction of **Appendix E.1**.

The next analysis shows that the proposed RCMP preserves structural invariance and refines the expressive power of the original SIL model.

Proposition 2. *Under **Assumptions B1-B4**, the following hold for RCMP.*

I. *The initialization $\widetilde{\mathrm{INIT}}_1(u, q, r)$ has relation invariance. Consequently, RCMP computes relation invariants in the relation encoder and link invariants in the entity encoder.*

II. *Let*
$$\widetilde{\mathrm{col}}_r^{(0)}(r_j|e_u, r_q) = \chi\Big(\widetilde{\mathrm{INIT}}_1(r_j|e_u, r_q)\Big),$$
where χ is injective, and let $(\widetilde{\mathrm{col}}_r, \widetilde{\mathrm{col}}_e)$ be the two-stage WL test of RCMP, while keeping all update rules unchanged. Then for all $n, \ell \geq 0$, we have
$$\widetilde{\mathrm{col}}_r^{(n)}(r_j|e_u, r_q) = \widetilde{\mathrm{col}}_r^{(n)}(r'_j|e'_u, r'_q) \Longrightarrow \mathrm{col}_r^{(n)}(r_j|r_q) = \mathrm{col}_r^{(n)}(r'_j|r'_q)$$
and
$$\widetilde{\mathrm{col}}_e^{(\ell)}(e_v|e_u, r_q) = \widetilde{\mathrm{col}}_e^{(\ell)}(e'_v|e'_u, r'_q) \Longrightarrow \mathrm{col}_e^{(\ell)}(e_v|e_u, r_q) = \mathrm{col}_e^{(\ell)}(e'_v|e'_u, r'_q).$$
Hence RCMP is at least as expressive as the original SIL model.

III. *Assume there exist a query $< e_u, r_q, ? >$ and two relations $r_j, r_s \in \mathcal{R}$ such that*
$$\mathrm{col}_r^{(n)}(r_j, r_q) = \mathrm{col}_r^{(n)}(r_s, r_q) \quad \textit{for all } n \geq 0,$$
but
$$\widetilde{\mathrm{INIT}}_1(r_j|e_u, r_q) \neq \widetilde{\mathrm{INIT}}_1(r_s|e_u, r_q).$$
Then the refinement is strict, that is, there exists a KG and two query links that cannot be separated by any instance of SIL, but can be separated by some instance of RCMP.

Proof. We prove the three claims as follows.

I. $\widetilde{\mathrm{INIT}}_1$ is a relation invariant. By **Assumption B1**, $\boldsymbol{R}_{|q}$ and $\boldsymbol{E}_{|u}$ depend only on KG structure, not on concrete entity or relation names. Since the prompt $\Pi(e_u, r_q)$ is formed only by these structural tokens and a fixed system prompt, **Assumption B2** implies that for every graph isomorphism $h = (\pi, \phi) : \mathcal{G} \to \mathcal{G}'$, we have
$$\boldsymbol{h}_{z|\Pi(e_u, r_q)} = \boldsymbol{h}_{z|\Pi(\pi(e_u), \phi(r_q))} \qquad \text{and} \qquad r_{z|\Pi(\pi(e_u), \phi(r_q))} = \phi(r_{z|\Pi(e_u, r_q)})$$
for all decoding steps z. Therefore, for every relation $r_j \in \mathcal{R}$, the accumulated rule signal satisfies
$$\boldsymbol{x}(r_j|e_u, r_q) = \boldsymbol{x}(\phi(r_j)|\pi(e_u), \phi(r_q)).$$
Since the query-anchor case $r_j = r_q$ is mapped to $\phi(r_j) = \phi(r_q)$ under the same isomorphism, the initialization satisfies
$$\widetilde{\mathrm{INIT}}_1(r_j|e_u, r_q) = \widetilde{\mathrm{INIT}}_1(\phi(r_j)|\pi(e_u), \phi(r_q)).$$
Hence $\widetilde{\mathrm{INIT}}_1$ is a relation invariant.

Now the relation encoder follows exactly the same inductive argument used in the main manuscript. Once the initialization $\widetilde{\mathrm{INIT}}_1$ is a relation invariant, every subsequent layer of the relation encoder remains a relation invariant because message, aggregation, and update only combine invariant quantities over the relational graph $\mathcal{G}_r$. Likewise, the entity encoder receives invariant relation representations and uses the same message-passing structure as the original SIL model, so it computes link invariants.

II. RCMP is at least as expressive as the original SIL model. We define the first-stage colors of RCMP by

$$\widetilde{\mathrm{col}}_r^{(0)}(r_j|e_u, r_q) = \chi\Big(\widetilde{\mathrm{INIT}}_1(r_j|e_u, r_q)\Big),$$

Then, for $n \geq 0$, we use the same first-stage update as in **Appendix E.1**, but with the richer initial coloring:

$$\widetilde{\mathrm{col}}_r^{(n+1)}(r_j|e_u, r_q) = \mathrm{HASH}\Big(\widetilde{\mathrm{col}}_r^{(n)}(r_j|e_u, r_q),\ \{\!\{(\{(\widetilde{\mathrm{col}}_r^{(n)}(r_s|e_u, r_q), p_s) \mid (r_s, p_s) \in \mathcal{N}_{r^*}(r_j)\}, r^*) \mid r^* \in \mathcal{R}^*\}\!\}\Big).$$

Similarly, define the second-stage colors by

$$\widetilde{\mathrm{col}}_e^{(0)}(e_v|e_u, r_q) = \mathbb{I}(e_v = e_u) \cdot \widetilde{\mathrm{col}}_r^{(N)}(r_q|e_u, r_q),$$

and for $\ell \geq 0$,

$$\widetilde{\mathrm{col}}_e^{(\ell+1)}(e_v|e_u, r_q) = \mathrm{HASH}\Big(\widetilde{\mathrm{col}}_e^{(\ell)}(e_v|e_u, r_q), \{\!\{(\widetilde{\mathrm{col}}_e^{(\ell)}(e_w|e_u, r_q), \widetilde{\mathrm{col}}_r^{(N)}(r_j|e_u, r_q)) \mid e_w \in \mathcal{N}_{r_j}(e_v),\ r_j \in \mathcal{R}\}\!\}\Big).$$

(1) We first prove by induction on n that

$$\widetilde{\mathrm{col}}_r^{(n)}(r_j|e_u, r_q) = \widetilde{\mathrm{col}}_r^{(n)}(r'_j|e'_u, r'_q) \Longrightarrow \mathrm{col}_r^{(n)}(r_j|r_q) = \mathrm{col}_r^{(n)}(r'_j|r'_q).$$

For $n = 0$, injectivity of χ gives $\widetilde{\mathrm{INIT}}_1(r_j|e_u, r_q) = \widetilde{\mathrm{INIT}}_1(r'_j|e'_u, r'_q)$. According to **Assumption B3**, such an equation preserves whether the anchor contribution $\mathbb{I}(r_j = r_q) \cdot \mathbf{1}$ is present. Therefore, $\mathbb{I}(r_j = r_q) = \mathbb{I}(r'_j = r'_q)$, which is exactly

$$\mathrm{col}_r^{(0)}(r_j|r_q) = \mathrm{col}_r^{(0)}(r'_j|r'_q).$$

Assume the implication holds at layer n and suppose

$$\widetilde{\mathrm{col}}_r^{(n+1)}(r_j|e_u, r_q) = \widetilde{\mathrm{col}}_r^{(n+1)}(r'_j|e'_u, r'_q).$$

Since HASH is injective, both the previous colors and the corresponding position-aware neighbor multisets of SIL must agree. Applying the induction hypothesis to all first-stage colors appearing inside those multisets shows that the original colors in **Appendix E.1** also agree. Therefore the multisets defining $\mathrm{col}_r^{(n+1)}(r_j|r_q)$ and $\mathrm{col}_r^{(n+1)}(r'_j|r'_q)$ coincide, thus

$$\mathrm{col}_r^{(n+1)}(r_j|r_q) = \mathrm{col}_r^{(n+1)}(r'_j|r'_q)$$

is proven.

(2) We next prove by induction on ℓ that

$$\widetilde{\mathrm{col}}_e^{(\ell)}(e_v|e_u, r_q) = \widetilde{\mathrm{col}}_e^{(\ell)}(e'_v|e'_u, r'_q) \Longrightarrow \mathrm{col}_e^{(\ell)}(e_v|e_u, r_q) = \mathrm{col}_e^{(\ell)}(e'_v|e'_u, r'_q).$$

For $\ell = 0$, equality of the RCMP colors implies

$$\mathbb{I}(e_v = e_u) = \mathbb{I}(e'_v = e'_u) \qquad \text{and} \qquad \widetilde{\mathrm{col}}_r^{(N)}(r_j|e_u, r_q) = \widetilde{\mathrm{col}}_r^{(N)}(r'_j|e'_u, r'_q).$$

By the first-stage result proved above, we obtain

$$\mathrm{col}_r^{(N)}(r_q|r_q) = \mathrm{col}_r^{(N)}(r'_q|r'_q),$$

hence

$$\mathrm{col}_e^{(0)}(e_v|e_u, r_q) = \mathrm{col}_e^{(0)}(e'_v|e'_u, r'_q).$$

Assume the implication holds at layer ℓ and suppose

$$\widetilde{\mathrm{col}}_e^{(\ell+1)}(e_v|e_u, r_q) = \widetilde{\mathrm{col}}_e^{(\ell+1)}(e'_v|e'_u, r'_q).$$

Injectivity of HASH yields equality of the previous second-stage colors of SIL and of the corresponding message multisets. By the induction hypothesis on ℓ and the first-stage implication already

proved, the original second-stage colors and original first-stage relation colors also agree. Therefore, the multisets defining $col_{F,T}^{(\ell+1)}(q(u,v))$ and $col_{F,T}^{(\ell+1)}(q'(u',v'))$ are identical, and the assumption

$$\mathrm{col}_e^{(\ell+1)}(e_v|e_u,r_q) = \mathrm{col}_e^{(\ell+1)}(e'_v|e'_u,r'_q)$$

is proven.

Therefore, the two-stage WL test of RCMP is a refinement of the original two-stage test of SIL on **Appendix E.1**. Under **Assumption B4**, the same constructive argument as in **Appendix E.1** applies to the present model, because only the initialization changes while the message-passing architecture over $\mathcal{G}_r$ and over the original KG remains unchanged. Therefore, the refined two-stage test characterizes RCMP, and the refinement of the WL tests implies that RCMP is at least as expressive as SIL.

III. Strictness. Assume there exist a query triplet $< e_u, r_q, ? >$ and two relations $r_j, r_s \in \mathcal{R}$ that satisfy

$$\mathrm{col}_r^{(n)}(r_j|r_q) = \mathrm{col}_r^{(n)}(r_s|r_q) \qquad \text{but} \qquad \widetilde{\mathrm{INIT}}_1(r_j|e_u,r_q) \neq \widetilde{\mathrm{INIT}}_1(r_s|e_u,r_q) \qquad \text{for all } n \geq 0.$$

We then construct a new KG $\mathcal{G}^\star$ with fresh entities $\{a,b,c,d,x,y\}$ and triplets $\{< x,r_j,b >, < y,r_s,d >\}$, and no other edges incident to b or d.

Consider two query triplets

$$< a,r_q,b > \qquad \text{and} \qquad < c,r_q,d >,$$

for the original SIL model, the incoming relation colors attached to b and d are the same because r_j and r_s are indistinguishable by the first-stage WL test. Moreover, the local entity neighborhoods of b and d are isomorphic because each target node has exactly one incoming edge from a fresh non-source node. Based on the above conditions, we can obtain

$$\mathrm{col}_e^{(\ell)}(b|a,r_q) = \mathrm{col}_e^{(\ell)}(d|c,r_q) \qquad \text{for all } \ell \geq 0.$$

For RCMP, we already have $\widetilde{\mathrm{col}}_r^{(0)}(r_j|e_u,r_q) \neq \widetilde{\mathrm{col}}_r^{(0)}(r_s|e_u,r_q)$. Therefore, after a first-stage update, the message multiset received by b contains

$$\big(\widetilde{\mathrm{col}}_e^{(0)}(x|a,r_q), \widetilde{\mathrm{col}}_r^{(N)}(r_j|a,r_q)\big),$$

whereas the one received by d contains

$$\big(\widetilde{\mathrm{col}}_e^{(0)}(y|c,r_q), \widetilde{\mathrm{col}}_r^{(N)}(r_s|c,r_q)\big).$$

Based on $\widetilde{\mathrm{col}}_e^{(0)}(e_v|e_u,r_q) = \mathbb{I}(e_v = e_u) \cdot \widetilde{\mathrm{col}}_r^{(N)}(r_q|e_u,r_q)$, the first components are equal, but the second components are different, so the multisets in the second-stage test are different. Finally, we can obtain

$$\widetilde{\mathrm{col}}_e^{(\ell)}(b|a,r_q) \neq \widetilde{\mathrm{col}}_e^{(\ell)}(d|c,r_q) \qquad \text{for all } \ell \geq 0.$$

Again using the constructive direction of **Assumption B4**, there exists an instance where RCMP can separate but SIL cannot. This proves the strict refinement. □

Discussion. The proposition identifies a novel perspective of expressiveness beyond changing the motif set $\mathcal{R}^*$ itself. The original theory strengthens SIL by enriching the relational hypergraph through richer motifs. Our construction keeps $\mathcal{R}^*$ fixed and instead strengthens the base coloring of the first-stage relation process by injecting a structure-only autoregressive rule signal. Therefore, the gain comes from refining the initial partition of relations rather than from adding new motifs. Moreover, the structure-only prompting design is crucial. The invariance statement depends on the equivariance of the generator with respect to isomorphisms, which would generally fail if one injected relation names, entity names, or arbitrary textual attributes into the prompt.

F Training Algorithm

Algorithm 1 provides a complete training process for SIRLM$_{\mathrm{PT}}$. In each training iteration, we first convert a query triplet into a instruction form consisting of textual and structural tokens (**Step 6**). Next, we obtain the structural representaions of entities and relations to construct a tokenizer for the query instruciton (**Steps 7 and 8**). We then obtain the structural relation memory and conduct in-context learning for the query instruciton (**Steps 9 and 10**), which provides the hidden states for rule generation (**Step 11**) and reasoning (**Step 12**). Finally, we using the reasoning loss to update the model parameters (**Step 16**) and return a pre-trained parameter set for downstream KGR tasks.

Algorithm 1 Pre-training framework of SIRLM

Input: KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$; relational graph $\mathcal{G}_r = (\mathcal{R}, \mathcal{R}^*, \mathcal{T}^*)$; trainable parameter set of SIRLM Ω; learning rate η; max training step s; batch size b.
Output: Optimized parameter set Ω.

$step = 0$
for $step < s$ **do**
  Randomly select b query triplets from $\mathcal{T}$ to form $\mathcal{T}_q$
  $\mathcal{L}_{total} = 0$
  for $< e_h, r_q, ? >$ in $\mathcal{T}_q$ **do**
    Construct the query instruction X of $< e_h, r_q, ? >$ according to Appendix A
    Obtain structural embeddings $\boldsymbol{R}_{|q}$ and $\boldsymbol{E}_{|h}$ according to Eqs (1) and (2), respectively
    Obtain the tokenizer TKN_{RG} according to Eq. (6)
    Obtain structural relation memory $\boldsymbol{R}_{\text{mem}}$ by Eq. (7)
    Convert X into token sequence using TKN_{RG} and conduct in-context learning with $\boldsymbol{R}_{\text{mem}}$ according to Eq. (8)
    Generate the rule body $\rho = \bigwedge_{z=1}^{\varepsilon} r_z$ with the hidden states $\{\boldsymbol{h}_{L+z}\}_{z=1}^{\varepsilon}$ using Eq. (10)
    Reason the missing entity in the query triplet according to Eqs. (11)-(13)
    Calculate the loss $\mathcal{L}_{\text{SFT}}$ using Eq. (14)
    $\mathcal{L}_{total} \leftarrow \mathcal{L}_{total} + \mathcal{L}_{\text{SFT}}$
  end for
  Optimize trainable parameters according to $\Omega \leftarrow \Omega - \eta \nabla(\mathcal{L}_{total})$
  $step \leftarrow step + 1$
end for
return Ω

Table 4: Comparison of KRLM and SIRLM in terms of TFLOPs, memory footprint, and wall-clock time under pre-training and fine-tuning settings.

Metric	Pre-training (3 transductive datasets)		Fine-tuning (FB15k237 v1)		Fine-tuning (FB15k237-25)	
	KRLM	**SIRLM$_{\text{PT}}$**	**KRLM**	**SIRLM$_{\text{SFT}}$**	**KRLM**	**SIRLM$_{\text{SFT}}$**
Avg. TFLOPs (forward propagation 100 steps)	3.3436±0.4540	**2.0397±0.3841**	3.2755±0.5208	**2.0683±0.0175**	3.3312±0.4859	**2.2287±0.0000**
Training memory	36.12 GB / GPU	**27.58 GB / GPU**	32.57 GB / GPU	**11.38 GB / GPU**	32.67 GB / GPU	**13.12 GB / GPU**
Wall-clock time	3h10m / epoch	**2h38m / epoch**	7m28s / epoch	**3m39s / epoch**	12m13s / epoch	**10m45s / epoch**

G Computational Complexity

The computational complexity of SIRLM consists of three parts. For the construction of the relational graph $\mathcal{G}^*$, since we need to dynamically construct it based on negative sampling during the training process, the time complexity of this part needs to be considered (Huang et al., 2025a). To construct motif edges in $\mathcal{G}^*$, we first need to compress $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{R}|\times|\mathcal{E}|}$ into two sparse matrices $\boldsymbol{A}_h \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{R}|}$ and $\boldsymbol{A}_t \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{E}|}$, with a complexity of $\mathcal{O}(|\mathcal{E}|^2|\mathcal{R}|)$. Next, the construction process of the four types of binary edges theoretically requires a complexity of $\mathcal{O}(|\mathcal{E}||\mathcal{R}|^2)$. In practical situations, we use the sparse operator $\text{spmm}(\cdot)$ to reduce the complexity of constructing binary edges to $\mathcal{O}(\text{nnz}(\boldsymbol{X}|\boldsymbol{X} \in \{\boldsymbol{A}_h, \boldsymbol{A}_t\}) \times \text{nnz}(\boldsymbol{Y}|\boldsymbol{Y} \in \{\boldsymbol{A}_h^{\text{T}}, \boldsymbol{A}_t^{\text{T}}\}))$, where $\text{nnz}(\cdot)$ is a operator finding the number of non-zero element in $\boldsymbol{X}$ and $\boldsymbol{Y}$. Similarly, the theoretical complexity and actual sparse computational complexity of constructing ternary edges are $\mathcal{O}(|\mathcal{E}|^2|\mathcal{R}|^2 + |\mathcal{E}||\mathcal{R}|^3)$ and $\mathcal{O}(\text{nnz}(\boldsymbol{X}|\boldsymbol{X} \in \{\boldsymbol{A}_h, \boldsymbol{A}_t\}) \times \text{nnz}(\boldsymbol{A}) \times \text{nnz}(\boldsymbol{Y}|\boldsymbol{Y} \in \{\boldsymbol{A}_h^{\text{T}}, \boldsymbol{A}_t^{\text{T}}\}))$, respectively

From the perspective of the KG tokenizer and RCMP reasoner, the time complexity is upper-bounded by the NBFNet executed on $\mathcal{G}$, as $|\mathcal{R}| \ll |\mathcal{E}|$. For each layer on NBFNet, the reasoning time complexity is calculated as $\mathcal{O}(|\mathcal{T}|d + |\mathcal{E}|d^2)$, where d is the embedding dimension. Therefore, for a N-layer NBFNet executed on $\mathcal{G}$, its overall time complexity is $\mathcal{O}(N(|\mathcal{T}|d + |\mathcal{E}|d^2))$. Furthermore, by using the efficient relational messaging kernel in the Pytorch-geometric library, the complexity of the NBFNet is optimized to $\mathcal{O}(N|\mathcal{E}|d)$ (Galkin et al., 2024).

Table 5: Element statistics of KGR dataset. "**Triplets**" represents the number of total triplets contained in a training/validation/testing graph. "**#Valid**" and "**#Test**" are the number of evaluation triplets in the validation and testing graph, respectively.

Type	Dataset	Training graph			Validation Graph				Testing Graph			
		Entities	Relations	Triplets	Entities	Relations	Triplets	#Valid	Entities	Relations	Triplets	#Test
Transductive	FB15k237 (Toutanova & Chen, 2015)	14541	237	272115	14541	237	272115	17535	14541	237	272115	20466
	CoDEx-M (Safavi & Koutra, 2020)	17050	51	185584	17050	51	185584	10310	17050	51	185584	10311
	WN18RR (Dettmers et al., 2018)	40943	11	86835	40943	11	86835	3034	40943	11	86835	3134
	NELL995 (Xiong et al., 2017)	74536	200	149678	74536	200	149678	543	74536	200	149678	2818
IndE	FB15k237 v1 (Teru et al., 2020)	1594	180	4245	1594	180	4245	489	1093	180	1993	411
	FB15k237 v2 (Teru et al., 2020)	2608	200	9739	2608	200	9739	1166	1660	200	4145	947
	FB15k237 v3 (Teru et al., 2020)	3668	215	17986	3668	215	17986	2194	2501	215	7406	1731
	FB15k237 v4 (Teru et al., 2020)	4707	219	27203	4707	219	27203	3352	3051	219	11714	2840
	WN18RR v1 (Teru et al., 2020)	2746	9	5410	2746	9	5410	630	922	9	1618	373
	WN18RR v2 (Teru et al., 2020)	6954	10	15262	6954	10	15262	1838	2757	10	4011	852
	WN18RR v3 (Teru et al., 2020)	12078	11	25901	12078	11	25901	3097	5084	11	6327	1143
	WN18RR v4 (Teru et al., 2020)	3861	9	7940	3861	9	7940	934	7084	9	12334	2823
	NELL995 v1 (Teru et al., 2020)	3103	14	4687	3103	14	4687	414	225	14	833	201
	NELL995 v2 (Teru et al., 2020)	2564	88	8219	2564	88	8219	922	2086	88	4586	935
	NELL995 v3 (Teru et al., 2020)	4647	142	16393	4647	142	16393	1851	3566	142	8048	1620
	NELL995 v4 (Teru et al., 2020)	2092	76	7546	2092	76	7546	876	2795	76	7073	1447
FullInd	FB15k237-25 (Lee et al., 2023)	5190	163	91571	4097	216	17147	5716	4097	216	17147	5716
	FB15k237-50 (Lee et al., 2023)	5190	153	85375	4445	205	11636	3879	4445	205	11636	3879
	FB15k237-75 (Lee et al., 2023)	4659	134	62809	2792	186	9316	3106	2792	186	9316	3106
	FB15k237-100 (Lee et al., 2023)	4659	134	62809	2624	77	6987	2329	2624	77	6987	2329
	NELL995-25 (Lee et al., 2023)	4396	106	17578	2146	120	2230	743	2146	120	2230	744
	NELL995-50 (Lee et al., 2023)	4396	106	17578	2335	119	2576	859	2335	119	2576	859
	NELL995-75 (Lee et al., 2023)	2607	96	11058	1578	116	1818	606	1578	116	1818	607
	NELL995-100 (Lee et al., 2023)	1258	55	7832	1709	53	2378	793	1709	53	2378	793
	Wikidata68K-25 (Lee et al., 2023)	12659	47	41873	3228	74	3391	1130	3228	74	3391	1131
	Wikidata68K-50 (Lee et al., 2023)	12022	72	82481	9328	93	9672	3224	9328	93	9672	3225
	Wikidata68K-75 (Lee et al., 2023)	6853	52	28741	2722	65	3430	1143	2722	65	3430	1144
	Wikidata68K-100 (Lee et al., 2023)	9784	67	49875	12136	37	13487	4496	12136	37	13487	4496
	MTDEA1-tax (Zhou et al., 2023)	10000	10	17178	10000	10	17178	1908	10000	9	16526	1834
	MTDEA1-health (Zhou et al., 2023)	10000	7	14371	10000	7	14371	1596	10000	7	14110	1566
	MTDEA2-org (Zhou et al., 2023)	10000	10	23233	10000	10	23233	2581	10000	11	21976	2441
	MTDEA2-sci (Zhou et al., 2023)	10000	16	16471	10000	16	16471	1830	10000	16	14852	1650
	MTDEA3-art (Zhou et al., 2023)	10000	45	27262	10000	45	27262	3026	10000	45	28023	3113
	MTDEA3-infra (Zhou et al., 2023)	10000	24	21990	10000	24	21990	2443	10000	27	21646	2405
	MTDEA4-sci (Zhou et al., 2023)	10000	42	12576	10000	42	12576	1397	10000	42	12516	1388
	MTDEA4-health (Zhou et al., 2023)	10000	21	15539	10000	21	15539	1725	10000	20	15337	1703

The complexity of the in-contex learning module with the sturctural relation memory can be divided into the self-attention matrix calculation ($\mathcal{O}(L^2F)$), the memory key-value calculation ($\mathcal{O}(LKd)$), and the final aggregation ($\mathcal{O}(L(L+K)F)$), where F is the hidden dimensions of LLM. Because $L \gg K$, the complexity upper bound of the in-contex learning module can be represented as $\mathcal{O}(L(L+K)F)$.

We further provide Table 4, which shows the TFLOPs in forward propagation, memory footprint, and wall-clock time of SIRLM$_{\text{PT}}$ and SIRLM$_{\text{SFT}}$ under the condition of batch_size = 4 per GPU × 4 GPUs. For the SFT paradigm, we include results on the largest inductive dataset (FB15k237-25) and the smallest inductive dataset (FB15k237 v1) to provide a boundary of the computational cost.

H Datasets

In our experiments, we conduct evaluations on 36 datasets. According to the overlap level between train KG $\mathcal{G}_{train} = (\mathcal{E}_{train}, \mathcal{R}_{train}, \mathcal{T}_{train})$ and test KG $\mathcal{G}_{test} = (\mathcal{E}_{test}, \mathcal{R}_{test}, \mathcal{T}_{test})$, these datasets can be divided into the following three categories:

- **Transductive** datasets that $\mathcal{E}_{test} = \mathcal{E}_{train}$ and $\mathcal{R}_{test} = \mathcal{R}_{train}$: FB15k-237 (Toutanova & Chen, 2015), WN18RR (Dettmers et al., 2018), CoDEx-M (Safavi & Koutra, 2020), and NELL995 (Xiong et al., 2017).
- **Ind**uctive **E**ntity (**IndE**) datasets that $\mathcal{E}_{test} \neq \mathcal{E}_{train}$ and $\mathcal{R}_{test} = \mathcal{R}_{train}$, including 12 datasets from GraIL (Teru et al., 2020) (FB15k237 V1, FB15k237 V2, FB15k237 V3, FB15k237 V4, WN18RR V1, WN18RR V2, WN18RR V3, WN18RR V4, NELL995 V1, NELL995 V2, NELL995 V3, and NELL995 V4).
- **Fully Ind**uctive (**FullInd**) datasets that $\mathcal{E}_{test} \neq \mathcal{E}_{train}$ and $\mathcal{R}_{test} \neq \mathcal{R}_{train}$, including 20 datasets from InGram (Lee et al., 2023) (FB15k237-25, FB15k237-50, FB15k237-75, FB15k237-100, NELL995-25, NELL995-50, NELL995-75, NELL995-100, Wikidata68K-25, Wikidata68K-50, Wikidata68K-75, and Wikidata68K-100) and MTDEA (Zhou et al., 2023) (MTDEA1-tax,



Figure 7: The length distribution of rules mined from various datasets.

MTDEA1-health, MTDEA2-org, MTDEA2-sci, MTDEA3-art, MTDEA3-infra, MTDEA4-sci, and MTDEA4-health).

These dataset are used to evaluate the model in the four training paradigms mentioned in Section 5.1. Table 5 provides detailed elemental statistics for these datasets.

Next, we discuss the rule mining approach in the data preprocessing stage. For each complete triplet in the training KG, we extract all closed paths within three hops. Due to the sparsity of the KG, some triplets may not have any closed paths of limited length that can be extracted. Figure 7 presents the proportion of triplets without rules (No rule) and the distribution of rules at different lengths across datasets. It can be observed that datasets such as WN18RR and the MTDEA1, MTDEA2, and MTDEA4 series exhibit a relatively high proportion of rule-less triplets. This is one of the reasons that the performance gains of SIRLM on these datasets, as shown in Tables 1, 8, and 9, are generally lower than on other datasets, *i.e.*, the difficulty in obtaining comprehensive rule-based contextual information from the KG.

After extracting the closed paths, we handle triplets without any rules by directly using the query relation of the triplet as the rule body. For the remaining triplets that contain at least one closed path, we further construct candidate rule bodies by sequentially aggregating the directed relation sequences along each closed path. We then count the occurrence frequency of all candidate rule bodies for each triplet and select the most frequent one as the rule to be generated for that triplet, *i.e.*, the masked component of the query instruction described in **Appendix A**.

Table 6: Hyperperameters of KRLM used in pre-training and end-to-end training from scratch.

Module	Component	Parameter
KG Tokenizer	NBFNet on $\mathcal{G}_r$	Layer number $N = 6$ Hidden dim $d = 64$ Message function $\text{MSG}(\cdot) = \text{DistMult}$ Aggregation function $\text{AGG}(\cdot) = \text{Sum}$ Updating function $\text{UP}(\cdot) = \text{nn.Linear}(128, 64)$ Motif Edge embeddings $\boldsymbol{R}^* = \text{nn.Embedding}(8, 64)$
	NBFNet on $\mathcal{G}$	Layer number $N = 6$ Hidden dim $d = 64$ Message function $\text{MSG}(\cdot) = \text{DistMult}$ Aggregation function $\text{AGG}(\cdot) = \text{Sum}$ Updating function $\text{UP}(\cdot) = \text{nn.Linear}(128, 64)$
	Up-scale layer $f_{\text{up}}(\cdot)$	$\text{nn.Linear}(64, 1536)$
	Score function $\mathcal{S}_{\text{KG}}(\cdot)$	$\text{nn.Linear}(128, 64)$ $\text{nn.ReLU}(\cdot)$ $\text{nn.Linear}(64, 1)$
In-context Learning Module	Qwen2.5-1.5b backbone	Default configuration
	LoRA configuration	$r = 64$ $\alpha = 32$ droupout $= 0.1$ target module = [gate_proj, up_proj, down_proj]
	Memory key layer $m_K(\cdot)$	$\text{nn.Linear}(64, 1536)$
	Memory value layer $m_V(\cdot)$	$\text{nn.Linear}(64, 1536)$
	Scale of structural relation memory	$K = 30$
Next-relation Predictor	Porject layer $f_{\boldsymbol{\Theta}}(\cdot)$	$\text{nn.Linear}(1600, 1600)$ $\text{nn.ReLU}(\cdot)$ $\text{nn.Linear}(1600, 1)$
RCMP Reasoner	NBFNet on $\mathcal{G}_r$	Layer number $N = 6$ Hidden dim $d = 64$ Message function $\text{MSG}(\cdot) = \text{DistMult}$ Aggregation function $\text{AGG}(\cdot) = \text{Sum}$ Updating function $\text{UP}(\cdot) = \text{nn.Linear}(128, 64)$ Motif Edge embeddings $\hat{\boldsymbol{R}}^* = \text{nn.Embedding}(8, 64)$
	NBFNet on $\mathcal{G}$	Layer number $N = 6$ Hidden dim $d = 64$ Message function $\text{MSG}(\cdot) = \text{DistMult}$ Aggregation function $\text{AGG}(\cdot) = \text{Sum}$ Updating function $\text{UP}(\cdot) = \text{nn.Linear}(128, 64)$
	Down-scale layer $f_{\text{down}}(\cdot)$	$\text{nn.Linear}(1536, 64)$
	Score function $\mathcal{S}_{\text{RCMP}}(\cdot)$	$\text{nn.Linear}(128, 64)$ $\text{nn.ReLU}(\cdot)$ $\text{nn.Linear}(64, 1)$
Training	Optimizer	AdamW
	Number of negative samples	512

I Experimental Hyperparameter Settings

In Section 5.2, we report four training paradigms of SIRLM, including End-to-End (E2E) training from scratch, Pre-Training (PT), SFT, and GRPO post-training. The hyperparameters of the model architecture under the four paradigms are uniformly set to the values in Table 6. In the PT paradigm, we set the learning rate to 1e-4 and use the AdamW optimizer with a 1% warm-up step. The batch size per GPU is 12. Each epoch consists of 5000 iterations and the training process is conducted for a total of 20 epochs. More detailed training hyperparameters for the other training paradigms are provided in Table 7.

J Details Experimental Analysis

J.1 Additional Analysis of Main Experiments

Tables 8 and 9 correspond to the detailed experimental results of each method in Table 1 on the IndE and FullInd datasets, respectively.

Traditional embedding-based methods rely on initializing fixed representations of entities and relations tailored to a specific KGR scenario.

Table 7: Detailed Training parameters, where b, η, and M represent batch size, learning rate, and GRPO sampling number, respectively. "all" means that each epoch needs to iterate through all training queries.

Datasets	$\text{SIRLM}_{\text{E2E}}$ (b, η, epoch, step)	$\text{SIRLM}_{\text{SFT}}$ (b, η, epoch, step)	$\text{SIRLM}_{\text{GRPO}}$ (b, η, M, epoch, step)
FB15k237	(12, 5e-4, 10, all)	(12, 1e-4, 3, all)	(8, 1e-5, 8, 1, all)
CoDEx-M	(24, 1e-4, 10, all)	(24, 5e-5, 3, all)	(8, 1e-5, 8, 1, all)
WN18RR	(12, 5e-4, 10, all)	(12, 1e-4, 3, all)	(8, 1e-5, 8, 3, all)
NELL995	(12, 5e-4, 10, all)	(12, 1e-4, 3, all)	(8, 1e-5, 8, 1, all)
FB15k237 v1	(24, 1e-4, 10, all)	(24, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
FB15k237 v2	(24, 1e-4, 10, all)	(24, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
FB15k237 v3	(24, 1e-4, 10, all)	(24, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
FB15k237 v4	(24, 1e-4, 5, all)	(24, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
WN18RR v1	(48, 1e-4, 10, all)	(48, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
WN18RR v2	(48, 1e-4, 10, all)	(48, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
WN18RR v3	(48, 1e-4, 20, all)	(48, 5e-5, 5, all)	(16, 1e-5, 8, 3, all)
WN18RR v4	(48, 1e-4, 10, all)	(48, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
NELL995 v1	(32, 1e-4, 5, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
NELL995 v2	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
NELL995 v3	(32, 1e-4, 5, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
NELL995 v4	(32, 1e-4, 5, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
FB15k237-25	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(8, 1e-5, 8, 3, all)
FB15k237-50	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(8, 1e-5, 8, 3, all)
FB15k237-75	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(8, 1e-5, 8, 3, all)
FB15k237-100	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(8, 1e-5, 8, 3, all)
NELL995-25	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(8, 1e-5, 8, 3, all)
NELL995-50	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(8, 1e-5, 8, 3, all)
NELL995-75	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(8, 1e-5, 8, 3, all)
NELL995-100	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(8, 1e-5, 8, 3, all)
Wikidata68K-25	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
Wikidata68K-50	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
Wikidata68K-75	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
Wikidata68K-100	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
MTDEA1-tax	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
MTDEA1-health	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
MTDEA2-org	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
MTDEA2-sci	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
MTDEA3-art	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
MTDEA3-infra	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
MTDEA4-sci	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)
MTDEA4-health	(32, 1e-4, 10, all)	(32, 5e-5, 3, all)	(16, 1e-5, 8, 3, all)

As a result, they cannot accommodate unseen entities and relations in the inference KG, making them unsuitable for inductive KGR tasks. Rule-based approaches, while capable of handling predictions involving unseen entities under a fixed set of relation types, fail to effectively generalize rule bodies when relations are dynamic, which prevents their application to FullInd KGR settings. In contrast, GNN-based methods are applicable across all KGR scenarios considered in our experiments. This advantage stems from their ability to learn structural context, enabling more structural pattern induction over the entire KG and allowing them to identify unfamiliar entities and relations through representations grounded in structural invariance.

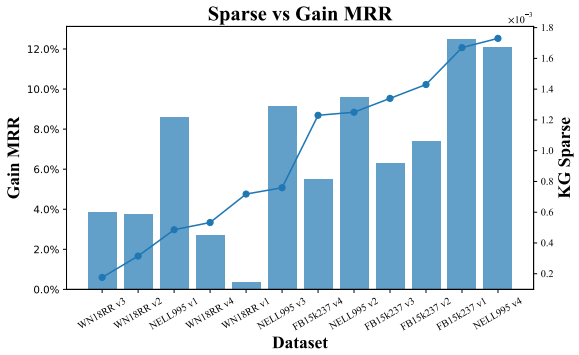


Figure 8: The correlation trend between KG sparsity and the MRR gain of SIRLM.

LLM-based methods leverage pre-trained knowledge and strong contextual reasoning capabilities to uncover latent facts in incomplete KGs and to recognize unseen entities and relations. For instance, ChatRule utilizes the natural language understanding ability of LLMs to filter candidate textualized KG rules and select those that best support reasoning over a query triplet, while MKGL and KRLM project explicit KG structures into the implicit parametric knowledge of LLMs and ultimately generate potential facts through next-entity prediction. However, these approaches lack faithfulness constraints on the implicit reasoning process of LLMs, making them susceptible to biases arising from misalignment between structural knowledge and parametric representations, which can lead to erroneous inference outcomes.

In contrast, our SIRLM enforces the generation of structurally grounded rules over the KG, mitigating the tendency of LLMs to overlook the underlying structural evidence in favor of pre-trained knowledge

Table 8: The overall performance of various methods on IndE datasets. The colored cells represent the best, second-best, and third-best values, respectively.

Methods	FB15k237 v1		FB15k237 v2		FB15k237 v3		FB15k237 v4	
	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑
NeuralLP	0.325	0.468	0.389	0.586	0.400	0.571	0.396	0.593
DRUM	0.333	0.474	0.395	0.595	0.402	0.571	0.410	0.593
NBFNet	0.422	0.574	0.514	0.685	0.476	0.637	0.453	0.627
RED-GNN	0.369	0.483	0.469	0.629	0.445	0.603	0.442	0.621
ULTRA	0.509	0.670	0.524	0.710	0.504	0.663	0.496	0.684
MOTIF	0.530	0.702	**0.557**	**0.744**	0.519	0.684	0.508	0.695
ChatRule†	0.448	0.546	0.484	0.628	0.471	0.653	0.457	0.598
MKGL†	0.475	0.595	0.508	0.681	0.486	0.643	0.471	0.645
KRLM†	0.511	0.668	0.527	0.712	0.520	0.681	0.505	0.693
$\text{SIRLM}_{\text{E2E}}$	0.548	**0.710**	**0.557**	**0.744**	**0.527**	**0.705**	**0.512**	**0.708**
SIRLM_{PT}	**0.552**	**0.718**	0.543	0.741	0.510	**0.702**	0.497	**0.705**
$\text{SIRLM}_{\text{SFT}}$	**0.558**	**0.718**	**0.555**	**0.747**	**0.529**	**0.708**	**0.510**	0.702
$\text{SIRLM}_{\text{GRPO}}$	**0.561**	**0.721**	**0.559**	**0.752**	**0.532**	**0.711**	**0.515**	**0.714**
Avg. gain⋆	*+12.50%*	*+14.50%*	*+7.38%*	*+8.87%*	*+6.28%*	*+7.70%*	*+5.52%*	*+7.52%*

Methods	WN18RR v1		WN18RR v2		WN18RR v3		WN18RR v4	
	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑
NeuralLP	0.649	0.772	0.635	0.749	0.361	0.476	0.628	0.706
DRUM	0.666	0.777	0.646	0.747	0.380	0.477	0.627	0.702
NBFNet	**0.741**	**0.826**	0.704	**0.798**	0.432	**0.568**	0.641	0.694
RED-GNN	0.701	0.799	0.690	0.780	0.427	0.524	0.651	0.721
ULTRA	0.685	0.793	0.679	0.779	0.411	0.546	0.614	0.720
MOTIF	0.703	**0.806**	0.680	0.781	**0.466**	**0.590**	**0.659**	0.733
ChatRule†	0.621	0.694	0.650	0.686	0.395	0.497	0.568	0.649
MKGL†	**0.746**	**0.822**	**0.712**	**0.799**	0.456	0.559	**0.664**	**0.741**
KRLM†	0.705	0.797	0.690	**0.791**	**0.457**	**0.590**	**0.662**	**0.737**
$\text{SIRLM}_{\text{E2E}}$	0.705	0.766	0.699	0.789	0.456	0.562	0.658	0.732
SIRLM_{PT}	0.668	0.744	0.670	0.766	0.433	0.549	0.627	0.693
$\text{SIRLM}_{\text{SFT}}$	**0.706**	0.778	**0.714**	**0.799**	**0.459**	**0.568**	**0.662**	**0.745**
$\text{SIRLM}_{\text{GRPO}}$	0.695	0.762	**0.708**	**0.791**	0.454	**0.566**	**0.662**	**0.741**
Avg. gain⋆	*+0.38%*	*+0.21%*	*+3.78%*	*+3.12%*	*+3.84%*	*+3.17%*	*+2.71%*	*+3.36%*

Methods	NELL995 v1		NELL995 v2		NELL995 v3		NELL995 v4	
	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑
NeuralLP	0.610	0.871	0.361	0.564	0.367	0.576	0.261	0.539
DRUM	0.628	0.873	0.365	0.540	0.375	0.577	0.270	0.531
NBFNet	0.648	0.862	0.421	0.599	0.462	0.578	0.404	0.588
RED-GNN	0.637	0.866	0.419	0.601	0.436	0.594	0.363	0.556
ULTRA	0.757	0.878	0.575	0.761	0.563	0.755	0.469	0.733
MOTIF	0.712	0.873	0.566	0.765	0.580	0.764	0.507	0.740
ChatRule†	0.719	0.806	0.529	0.690	0.552	0.732	0.528	0.741
MKGL†	**0.749**	**0.886**	0.570	0.767	0.571	0.759	0.525	0.749
KRLM†	0.679	**0.898**	0.568	0.761	0.561	0.755	**0.554**	0.762
$\text{SIRLM}_{\text{E2E}}$	0.680	0.871	**0.580**	**0.769**	**0.585**	**0.769**	**0.547**	**0.767**
SIRLM_{PT}	**0.762**	**0.908**	**0.579**	**0.787**	**0.583**	**0.775**	0.539	0.764
$\text{SIRLM}_{\text{SFT}}$	**0.768**	0.879	**0.582**	**0.789**	**0.588**	**0.779**	**0.552**	**0.771**
$\text{SIRLM}_{\text{GRPO}}$	0.742	0.880	0.572	0.768	0.576	0.757	0.543	**0.769**
Avg. gain⋆	*+8.59%*	*+3.99%*	*+9.60%*	*+11.70%*	*+9.17%*	*+10.23%*	*+12.08%*	*+11.11%*

† We reproduce the experimental metrics of ChatRule, MKGL, and KRLM.
⋆ We calculate the average gain of the optimal value in the four training modes of SIRLM compared to all baselines.

Table 9: The overall performance of various methods on FullInd datasets. The colored cells represent the best, second-best, and third-best values, respectively.

Methods	FB15k237-25		FB15k237-50		FB15k237-75		FB15k237-100	
	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑
NBFNet	0.224	0.410	0.130	0.259	0.089	0.166	0.072	0.154
RED-GNN	0.145	0.284	0.129	0.251	0.107	0.201	0.121	0.263
ULTRA	0.383	0.635	0.334	0.538	0.400	0.598	0.444	0.643
MOTIF	**0.388**	0.635	**0.340**	**0.544**	0.399	0.607	0.439	0.642
ChatRule†	0.336	0.602	0.289	0.514	0.250	0.507	0.361	0.554
KRLM†	**0.398**	**0.640**	**0.345**	**0.552**	**0.414**	**0.620**	**0.455**	**0.655**
$\text{SIRLM}_{\text{E2E}}$	0.386	0.635	**0.345**	0.540	**0.411**	**0.623**	0.444	0.646
SIRLM_{PT}	0.376	0.631	0.315	0.534	0.396	0.613	0.442	**0.655**
$\text{SIRLM}_{\text{SFT}}$	**0.388**	**0.647**	**0.345**	0.541	0.400	0.610	**0.448**	**0.648**
$\text{SIRLM}_{\text{GRPO}}$	**0.396**	**0.649**	**0.347**	**0.546**	**0.420**	**0.626**	**0.449**	**0.656**
Avg. gain⋆	*+8.37%*	*+11.47%*	*+8.58%*	*+10.30%*	*+14.35%*	*+17.62%*	*+13.37%*	*+17.08%*

Methods	NELL995-25		NELL995-50		NELL995-75		NELL995-100	
	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑
NBFNet	0.283	0.417	0.225	0.346	0.137	0.255	0.096	0.199
RED-GNN	0.214	0.266	0.179	0.115	0.203	0.353	0.212	0.385
ULTRA	**0.407**	0.596	**0.418**	**0.595**	**0.374**	**0.570**	0.458	**0.684**
MOTIF	0.390	0.580	**0.414**	0.573	0.360	0.548	0.464	**0.682**
ChatRule†	0.359	0.564	0.368	0.540	0.302	0.521	0.444	0.608
KRLM†	0.401	0.596	**0.432**	**0.598**	0.367	0.559	**0.489**	**0.688**
$\text{SIRLM}_{\text{E2E}}$	**0.407**	**0.601**	0.413	**0.595**	**0.371**	**0.570**	**0.477**	0.666
SIRLM_{PT}	0.394	0.593	0.394	**0.581**	0.355	0.550	**0.472**	0.668
$\text{SIRLM}_{\text{SFT}}$	**0.412**	**0.613**	**0.418**	**0.598**	**0.379**	**0.578**	**0.489**	0.667
$\text{SIRLM}_{\text{GRPO}}$	**0.410**	**0.607**	0.382	0.567	0.368	**0.560**	**0.477**	0.668
Avg. gain⋆	*+6.97%*	*+10.98%*	*+7.87%*	*+13.68%*	*+8.85%*	*+11.03%*	*+12.85%*	*+12.70%*

Methods	Wikidata68K-25		Wikidata68K-50		Wikidata68K-75		Wikidata68K-100	
	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑
NBFNet	0.154	0.301	0.062	0.105	0.072	0.172	0.014	0.026
RED-GNN	0.170	0.263	0.058	0.093	0.172	0.290	0.096	0.136
ULTRA	0.321	0.535	0.140	0.280	0.380	0.530	0.168	0.286
MOTIF	0.317	0.505	0.160	0.304	0.371	0.535	0.173	0.284
ChatRule†	0.256	0.372	0.090	0.126	0.266	0.483	0.107	0.196
KRLM†	**0.332**	**0.550**	0.168	**0.328**	0.384	0.538	**0.189**	**0.313**
$\text{SIRLM}_{\text{E2E}}$	**0.322**	**0.536**	**0.173**	**0.329**	**0.406**	**0.568**	**0.186**	0.304
SIRLM_{PT}	0.303	0.497	0.166	0.309	**0.399**	**0.556**	**0.189**	0.300
$\text{SIRLM}_{\text{SFT}}$	**0.328**	**0.541**	**0.171**	**0.330**	**0.412**	**0.566**	**0.191**	**0.316**
$\text{SIRLM}_{\text{GRPO}}$	0.314	0.511	**0.170**	0.315	0.385	0.550	0.185	**0.307**
Avg. gain⋆	*+6.97%*	*+12.00%*	*+5.80%*	*+12.40%*	*+13.78%*	*+14.33%*	*+6.65%*	*+10.92%*

Methods	MTDEA1-tax		MTDEA1-health		MTDEA2-org		MTDEA2-sci	
	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑
NBFNet	0.113	0.315	0.122	0.339	**0.109**	**0.292**	0.091	0.235
ULTRA	0.330	0.459	0.380	0.467	**0.104**	**0.170**	0.311	0.451
MOTIF	**0.416**	0.522	**0.385**	**0.473**	**0.106**	**0.170**	0.326	0.520
ChatRule†	0.300	0.388	0.352	0.427	0.101	0.140	0.343	0.468
KRLM†	0.378	0.508	0.376	0.457	0.098	0.140	0.361	0.526
$\text{SIRLM}_{\text{E2E}}$	0.399	**0.528**	**0.390**	**0.475**	0.101	0.142	0.385	0.530
SIRLM_{PT}	0.395	0.523	0.383	0.470	0.083	0.136	**0.390**	**0.532**
$\text{SIRLM}_{\text{SFT}}$	**0.407**	**0.533**	**0.404**	**0.481**	**0.109**	**0.145**	**0.410**	**0.548**
$\text{SIRLM}_{\text{GRPO}}$	**0.401**	**0.530**	0.384	**0.473**	0.097	0.140	**0.399**	**0.538**
Avg. gain⋆	*+9.96%*	*+9.46%*	*+8.10%*	*+4.84%*	*+0.54%*	*-3.74%*	*+12.36%*	*+10.80%*

Methods	MTDEA3-art		MTDEA3-infra		MTDEA4-sci		MTDEA4-health	
	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑	MRR↑	Hit10↑
NBFNet	0.099	0.257	0.137	0.357	0.094	0.246	0.124	0.328
ULTRA	0.306	0.473	0.657	0.807	0.303	0.478	0.704	**0.785**
MOTIF	0.315	0.469	**0.683**	0.827	0.309	0.483	0.703	**0.787**
ChatRule†	0.267	0.432	0.659	0.811	0.268	0.449	0.658	0.740
KRLM†	0.315	0.468	0.670	0.811	0.303	0.471	0.695	0.776
$\text{SIRLM}_{\text{E2E}}$	**0.328**	**0.499**	**0.686**	**0.834**	0.311	0.489	**0.710**	**0.789**
SIRLM_{PT}	**0.324**	0.492	0.677	0.827	**0.317**	**0.497**	**0.707**	**0.787**
$\text{SIRLM}_{\text{SFT}}$	**0.328**	**0.493**	**0.688**	**0.839**	**0.330**	**0.526**	0.701	**0.787**
$\text{SIRLM}_{\text{GRPO}}$	**0.333**	**0.510**	0.676	**0.830**	**0.321**	**0.514**	**0.706**	**0.789**
Avg. gain⋆	*+7.26%*	*+9.02%*	*+12.68%*	*+11.64%*	*+7.46%*	*+10.06%*	*+13.32%*	*+10.58%*

† We reproduce the experimental metrics of ChatRule and KRLM.
⋆ We calculate the average gain of the optimal value in the four training modes of SIRLM compared to all baselines.

Table 10: The overall performance of ablation variants on different datasets, where the MRR and Hit10 in the IndE and FullInd scenarios are summarized as average values. The colored cells represent the best, second-best, and third-best values, respectively.

Methods	FB15k237		CodeX-M		WN18RR		NELL995		12 IndE Datasets		20 FullInd Datasets	
	MRR	Hit10	MRR	Hit10	MRR	Hit10	MRR	Hit10	MRR	Hit10	MRR	Hit10
Full Model	**0.408**	**0.593**	**0.367**	**0.522**	**0.461**	**0.556**	**0.520**	**0.638**	**0.580**	**0.738**	**0.374**	**0.538**
w/o **MMQI**	**0.389**	**0.579**	**0.360**	**0.499**	**0.458**	**0.525**	**0.514**	**0.627**	**0.570**	**0.721**	**0.364**	**0.522**
w/o **SRM**	**0.358**	**0.566**	**0.348**	**0.478**	**0.455**	0.526	0.488	0.606	**0.558**	**0.709**	**0.348**	**0.508**
w/o **RCMP**	0.352	0.538	0.339	0.422	0.404	0.500	**0.501**	**0.611**	0.540	0.685	0.344	0.502

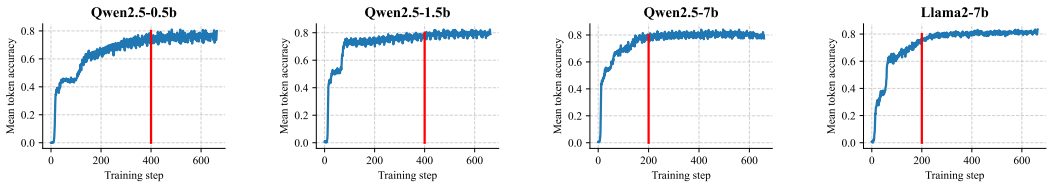


Figure 9: Convergence of rule generation accuracy for SIRLM with different LLMs on WN18RR v1.

when generating answers. Furthermore, we observe a correlation between the performance gains of SIRLM and the sparsity of the KG. Based on the results in Tables 8 and 9, Figure 8 illustrates this trend. Notably, SIRLM performs slightly worse on sparse KGs (*e.g.*, the WN18RR series dataset) than on dense ones, which highlights an inherent limitation of structural rule-based reasoning under sparse conditions and points to an important direction for future research.

J.2 Details Ablation Analysis

Section 5.3 analyzes the ablation components of SIRLM. To verify the universality of each component, we chose to perform ablation variants in SIRLM_{PT} training mode. We first provide the design details of each ablation variant.

- **-MMQI**. This variant only provides the textual instructions of query triplets, without including structural representations of entities and relations. Following MKGL (Guo et al., 2024), such instruction sequences contain textual strings of entities/relations along with their corresponding textual explanations. During tokenization, we use the principal neighborhood aggregation method (Corso et al., 2020) to aggregate the text token sequences of entity/relation into word-level token embeddings, which are then used for subsequent rule generation and reasoning. Since the instructions do not include structural representations, the tokenizer in Eq. (6) degenerates into TKN_{LLM}. The rest of the SIRLM architecture remains unchanged.
- **-SRM**. This variant removes the structural relation memory in the in-context learning module, along with its corresponding linear layers ($m_K(\cdot)$ and $m_V(\cdot)$) in Eq. (8). All other components remain consistent with the primary SIRLM.
- **-RCMP**. This variant removes $\{< r_z, f_{\text{down}}(\boldsymbol{h}_{L+z}) >\}_{z=1}^{\varepsilon}$ in Eq. (11), reverting it back to the original SIL module.

Table 10 provides the details ablation experimental results of each ablation variant.

J.3 Additional Analysis on Different LLM Backbones

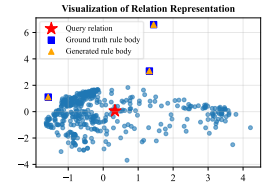
Section 5.5 presents the experimental results of SIRLM across four different LLM backbones. Overall, the results show a positive correlation between the parameter scale of the backbone and its reasoning performance. We attribute this to the fact that larger parameter sizes enable the model to more quickly fit the newly injected structured representation space.

To further illustrate this, Figure 9 reports the convergence of rule generation accuracy for the four LLM backbones on WN18RR v1. It is evident that the two 7B backbones (Qwen2.5-7b and Llama2-7b) approach convergence at around 200 steps, whereas Qwen2.5-0.5b and Qwen2.5-1.5b backbones require approximately 400 steps to reach a similar level of convergence.

J.4 Case Study and Error Analysis

To better understand the behavior of our SIRLM, we visualize several representative cases of rule generation in Figure 10. **Cases 1 and 2** is a successful example where the generated rule exactly matches the ground-truth rule, indicating that the model can accurately identify the correct multi-hop reasoning pattern. As a result, the correct entities "**Best Academy Picture Award**" and "**University of Texas at Austin**" appears in the top-ranked candidates with a relatively high confidence score.

Cases 3 and 4 demonstrates that an exact match to the ground-truth rule is not necessary for correct prediction. Although the generated rule deviates from the annotated reasoning path, it still leads to



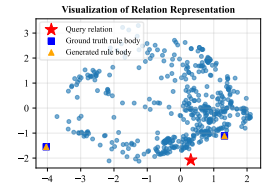
Case 1: <Howard Hughes, *award_nominations*, **Best Academy Picture Award**>

Ground truth rule body:
award_nominations=*nationality*→*inv_country*→*inv_nominated_for*

Generated rule body:
award_nominations=*nationality*→*inv_country*→*inv_nominated_for*

Top-5 candidates:

Best Academy Actress Award: 0.946
Best Academy Picture Award: 0.860
BAFTA Best Film Award : 0.546
Best Academy Director Award: 0.524
Golden Raspberry Award: 0.358



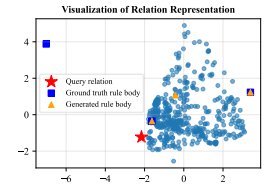
Case 2: <Mitch Pileggi, *gdp_nominal*, **University of Texas at Austin**>

Ground truth rule body:
gdp_nominal=*nationality*→*inv_students_graduates*

Generated rule body:
gdp_nominal=*nationality*→*inv_students_graduates*

Top-5 candidates:

University of Texas at Austin: 0.822
Princeton University: 0.801
Cornell University: 0.696
New York University: 0.683
Yale University : 0.643



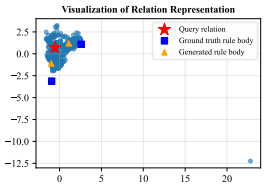
Case 3: <Bandai Namco Holdings, *headquarters_state_province*, **Tokyo**>

Ground truth rule body:
headquarters_state_province=*webpage_category*→*inv_webpage_category*→*headquarters_citytown*

Generated rule body:
headquarters_state_province=*webpage_category*→*inv_webpage_category*→*inv_contains*

Top-5 candidates:

Tokyo: 0.991
California: 0. 952
Minato: 0.851
Santa Clara: 0.736
Sapporo: 0.519



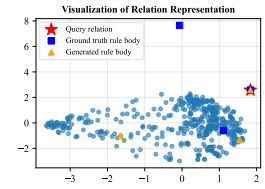
Case 4: <Nick Jonas, *inv_taxonomy*, **Columbia Records**>

Ground truth rule body:
inv_taxonomy=*profession*→*inv_profession*

Generated rule body:
inv_taxonomy=*inv_artists*→*artists*

Top-5 candidates:

Columbia Records: 0.961
Avex Trax: 0.869
Island Records: 0.628
RCA Records: 0.549
Atlantic Records: 0.517



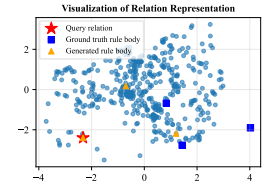
Case 5: <Columbia Pictures, *films_distributed*, **Hotel Transylvania**>

Ground truth rule body:
films_distributed=*films_distributed*→*release_region*→*inv_release_region*

Generated rule body:
films_distributed=*films_distributed*→*film_crew_role*→*inv_film_crew_role*

Top-5 candidates:

Chuck & Larry: 0.593
Happy Gilmore: 0.585
Halloween: 0.571
15 Minutes: 0.570
Bedtime Stories: 0.559



Case 6: <Allison Janney, *location*, **Dayton, Ohio**>

Ground truth rule body:
location=*award_nominations*→*nationality*→*contains*

Generated rule body:
location=*profession*→*inv_profession*→*location*

Top-5 candidates:

New York City: 0.796
Los Angeles: 0.758
Brooklyn: 0.639
London: 0.518
Santa Ana, California: 0.499

Figure 10: Case studies of rule generation. *Red*, *blue*, and *orange* denote the query relation, the atomic relations in the ground-truth rule, and the atomic relations in the predicted rule, respectively. **Green** denotes the target entities that need to be predicted.

the correct answers "**Tokyo**" and "**Columbia Records**". This suggests that the model can exploit alternative reasoning paths with similar structural semantics. In these case, the generated atomic relations (blue square) differ from the ground-truth atomic relations (orange triangle), but remain semantically similar in the representation space.

In contrast, **Cases 5 and 6** reveals a limitation of the method. The generated rule introduces irrelevant atomic relations that differ substantially from the ground truth, causing the reasoning process to deviate from the correct semantic direction and resulting in incorrect top-ranked candidates.

These observations indicate that the effectiveness of SIRLM critically depends on the quality of the generated rule body, that is, semantically accurate rules improve both interpretability and predictive performance, whereas semantically inconsistent rules can significantly degrade performance.

K Limitations and Future Work

SIRLM provides a novel modeling and training framework for LLM-based KGR research, effectively alleviating the problem of reasoning evidence perception drift caused by the knowledge representation gap between LLMs and KGs. However, SIRLM still has several potential limitations. We discuss these limitations as follows and provide potential future research directions.

Computational complexity. As shown in **Appendix G**, SIRLM needs to make real-time relational graph updates and message passing over the entire KG during training. Although sparse operators are adopted in practice to significantly reduce the computational cost of graph processing, the upper bound remains quadratic in the number of entities, which leads to a notable computational bottleneck for large-scale KGs. Therefore, in future work, we plan to introduce a query-driven evidence graph extraction mechanism (Huang et al., 2026), such that the graph structure processed by SIRLM is closely aligned with the query triplets. This would help eliminate the unnecessary computational overhead incurred by processing irrelevant KG context.

Sparse KG reasoning. The comprehensive analysis of Figures 7 and 8, as well as Tables 8 and 9, shows that there is still significant room for improvement in the reasoning performance of SIRLM on sparse KGs. We consider that this limitation is attributed to the fact that SIRLM is a rule-driven KGR framework. In sparse KGs, the diversity and completeness of structural rules are inherently limited, which in turn hinders the subsequent processes of rule learning and generation by LLMs. To address this issue, we plan to incorporate a soft rule mechanism (Qu et al., 2021) into the rule mining stage. Specifically, instead of relying solely on hard inductive methods based on observable closed-path patterns, we will jointly employ soft rule mining to construct approximate rules and corresponding soft labels for query triplets that lack closed-path evidence. These soft signals can serve as additional supervision for subsequent LLM-based rule generation. In this way, the contextual exploration space of LLMs over sparse KGs can be effectively expanded, thereby alleviating the limitations in rule learning caused by insufficient structural context.