CHAINGRAM: DISCOVERING A STRUCTURAL GRAMMAR FOR KNOWLEDGE GRAPH FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge graph foundation models (KGFMs) are emerging as a promising paradigm for transferable reasoning over knowledge graphs. By learning invariant patterns, they enable zero-shot inference on new KGs with unseen entities and relations. Existing KGFMs capture these patterns through relational interactions. However, these patterns are often implicitly encoded in learned representations, making them difficult to identify, organize, and reuse explicitly. Inspired by natural language grammar, we ask whether invariant patterns across KGs can be explicitly abstracted and organized into a transferable *structural grammar*. To this end, we propose CHAINGRAM, a chain-based KGFM that learns such a grammar from relational chains across KGs for query-specific reasoning. Specifically, we propose a hierarchical codebook encoder to learn word-level structural codes and chain-level prototypes from relational chains. These prototypes are further aggregated into a relation-specific structural grammar. A grammar-guided reasoner then selects relevant patterns based on query-specific chains and aggregates pattern-consistent evidence for entity prediction. Experimental results across 54 KGs demonstrate that CHAINGRAM achieves the best performance across all six evaluation settings, improving overall MRR over the strongest baseline by 2.82% under zero-shot inference and 5.84% with full fine-tuning. Our code is available at https://anonymous.4open.science/r/ChainGram-2857/.

1 INTRODUCTION

Knowledge graph foundation models (KGFMs) are emerging as a general framework for transferable reasoning over knowledge graphs. Instead of training a separate model for each graph, KGFMs learn transferable reasoning capabilities that generalize to unseen entities, relations, and entirely new KGs (Galkin et al., 2024; Arun et al., 2025).These capabilities are critical in real-world applications, enabling a single pretrained KGFM to uncover missing knowledge in new KGs without graph-specific retraining, supporting applications such as drug discovery, materials science, and biomedical research (Huang et al., 2024; Lopez et al., 2025; Ding et al., 2026). These capabilities position KGFMs as a promising foundation paradigm for knowledge graph reasoning, applicable across diverse domains and tasks (Wang et al., 2025).

Existing KGFMs have explored different ways to enhance transferable capabilities in KGs. Early studies model relational dependencies through auxiliary relation graphs, capturing how relations co-occur or interact with each other (Galkin et al., 2024; Cui et al., 2024; Zhang et al., 2025). Later approaches incorporate higher-level structural abstractions, such as motifs or graphlets, to capture richer local topology across graphs (Huang et al., 2025a; Amouzouvi et al., 2026). Together, these efforts demonstrate strong cross-graph transferability, implying the existence of **invariant patterns** shared across KGs with different relational structures and topological organizations.

However, these invariant patterns are often *implicitly encoded in* graph-level representations, making them difficult to identify and reuse explicitly. Since these representations are learned through graph-level aggregation over individual KGs, different patterns are jointly encoded into latent representations, making the invariant patterns difficult to explicitly identify and reuse. The central issue is therefore how to explicitly identify these patterns across KGs and leverage them for reasoning.

Inspired by natural language grammar, we ask a simple yet fundamental question: Is there a transferable *structural grammar* that explicitly captures the invariant patterns shared across KGs? In

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.

language, grammar constrains invariant patterns, e.g., subject–verb–object ordering that applies across word choices and facilitates language model generalization (Lake & Baroni, 2018; Press et al., 2023). Fig. 1 illustrates a similar phenomenon in KGs: the geographic KG connects Paris to Europe through both the two-hop chain *capital–part of* and the one-hop chain *located in*. The same chain-based structure recurs in the biomedical and sports KGs despite differences in entities and relations. By explicitly representing the sequential composition of relations, **Relational Chains** provide a natural basis for structural grammar. They capture multi-hop relation composition, analogous to how grammar combines basic units into larger structures (Wei et al., 2022). However, learning and applying this chain-based structural grammar for cross-graph reasoning pose two challenges:

Challenge 1: How to discover the transferable structural grammar from relational chains? Graph structures contain invariant patterns at different levels. The challenge is to capture these patterns through relational chains and explicitly encode them into a transferable structural grammar.

Challenge 2: How to match the learned structural grammar with query-relevant evidence for reasoning? Given a query, chains rooted at query entity are not equally relevant to reasoning. The challenge is to match these chains against the learned structural grammar to identify relevant evidence.

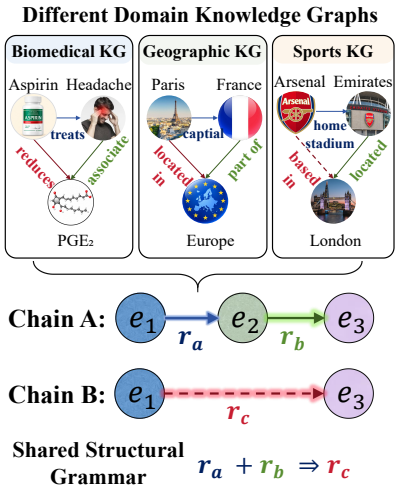


Figure 1: Shared structural language across KGs with different identities.

To address these challenges, we propose CHAINGRAM, a chain-based KGFM that explicitly captures invariant patterns via a transferable structural grammar. For the first challenge, we propose a hierarchical codebook encoder that progressively abstracts invariant patterns from local to global levels, yielding a structural vocabulary and transferable structural grammar over relational chains. For the second challenge, CHAINGRAM employs a grammar-guided reasoner that bridges abstract structural grammar and query-specific evidence through chain matching, enabling relevance-guided aggregation for entity prediction. In summary, the main contributions of our work are as follows:

- **Structural Grammar Framework.** We propose CHAINGRAM, a chain-based structural grammar framework that explicitly captures invariant patterns for cross-KG generalization.
- **Hierarchical Pattern Codebook.** We propose a hierarchical codebook encoder that captures invariant patterns through a relation-chain-grammar hierarchy, organizing them into a structural vocabulary and a transferable structural grammar.
- **Grammar-Guided Reasoning.** We develop a grammar-guided reasoner that selects relevant patterns based on query-specific chains and aggregates pattern evidence for entity prediction.

Extensive experiments across 54 KGs show that CHAINGRAM achieves the best performance, improving overall MRR by 2.82% in zero-shot inference and 5.84% with full fine-tuning. Further analyses show that the learned structural grammar captures invariant patterns reusable across KGs.

2 PRELIMINARIES

Knowledge Graph Reasoning. A knowledge graph (KG) is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$, where $\mathcal{V}$ is the set of entities, $\mathcal{R}$ is the set of relation types, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ is the set of observed facts. Each fact is represented as a triple (h, r, t), where $h, t \in \mathcal{V}$ and $r \in \mathcal{R}$. Given a query $(h, r, ?)$ or $(?, r, t)$, the goal of link prediction is to rank candidate entities according to the plausibility of the completed triple. To unify notation, we introduce an auxiliary inverse relation r^{-1} for each $r \in \mathcal{R}$, so that a head-prediction query $(?, r, t)$ can be rewritten as $(t, r^{-1}, ?)$. We therefore express all queries as $q = (s_q, r_q, ?)$. A model with parameters θ assigns each candidate entity $v \in \mathcal{V}$ a score

$$s(v \mid q) = s_\theta(\mathcal{G}_{\text{inf}}, s_q, r_q, v), \tag{1}$$

where $\mathcal{G}_{\text{inf}}$ is the observed graph available at inference time (defined below), and higher scores indicate greater plausibility of the completed triple.

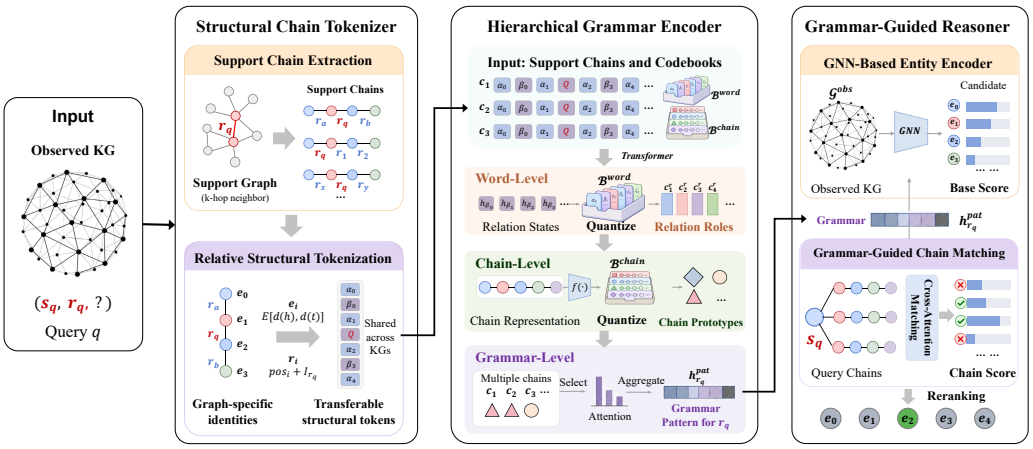


Figure 2: The overall architecture of the proposed CHAINGRAM.

Knowledge Graph Foundation Models. A KGFM is pretrained on a collection of source KGs $\mathbb{G}_{\text{src}} = \{\mathcal{G}_1, \ldots, \mathcal{G}_S\}$ and evaluated on target datasets not used for pretraining. In *zero-shot* evaluation, the pretrained parameters remain fixed, while the model uses the observed target graph as reasoning context; in *fine-tuning*, the KGFM is additionally adapted using the target dataset's training split. Following existing KGFM studies (Galkin et al., 2024; Cui et al., 2024), we categorize target datasets according to the relationship between the observed inference graph $\mathcal{G}_{\text{inf}}$ and the target KG $\mathcal{G}_{\text{tar}}$. In the *transductive* setting, entity and relation vocabularies are shared between training and inference graphs. In the *inductive-e* setting, inference involves unseen entities. The *inductive-e,r* setting additionally allows unseen relation types. Detailed definitions are provided in Appendix B.

Structural Grammar. We define a structural grammar as a mapping from relational chains to a shared invariant pattern space:

$$\mathcal{G} : \mathcal{C} \to \mathcal{Z},$$

where $\mathcal{C}$ denotes relational chains and $\mathcal{Z}$ denotes a transferable space of invariant patterns shared across KGs. $\mathcal{G}$ captures invariant relational compositions by abstracting local structural roles into reusable chain-level patterns. In CHAINGRAM, this mapping is parameterized hierarchically by local structural codes, chain prototypes, and relation-specific grammar patterns.

3 METHODOLOGY

In this section, we introduce CHAINGRAM, a chain-based KGFM that explicitly captures *invariant patterns* for transferable reasoning. As illustrated in Fig. 2, CHAINGRAM learns structural grammar through three modules: Structural Chain Tokenizer extracts support chains and converts them into query-relative structural tokens, Hierarchical Grammar Encoder abstracts them into invariant patterns, and Grammar-Guided Reasoner matches these patterns with query-specific chain evidence.

3.1 STRUCTURAL CHAIN TOKENIZER

To explicitly model transferable reasoning structures, we propose the **Structural Chain Tokenizer**, which derives transferable structural tokens from the support contexts of the query relation r_q. It contains two stages: *Support Chain Extraction* and *Relative Structural Tokenization*.

Support Chain Extraction. To get the graph context associated with query relation r_q, we construct local support contexts and extract support chains from them. Specifically, following KG-ICL (Cui et al., 2024), we sample a set of support facts $\mathcal{S}_{r_q} = \{c_i = (u_i, r_q, v_i)\}_{i=1}^{M}$ based on r_q. For each support fact c_i, we construct a support-specific prompt graph:

$$\mathcal{G}_i^{\text{pmt}} = \mathcal{G}\big[\mathcal{N}_k(u_i) \cup \mathcal{N}_k(v_i)\big]. \tag{2}$$

where $\mathcal{N}_k(\cdot)$ denotes the k-hop neighborhood. We then perform fixed-length random walks on each prompt graph. For each $\mathcal{G}_i^{\text{pmt}}$, we sample N support chains, with the j-th chain represented as $C_{i,j} = (e_{i,j,0}, r_{i,j,1}, \ldots, e_{i,j,L})$ where $e_{i,j,0} \in \{u_i, v_i\}$ is the start entity and L is the walk length.

Relative Structural Tokenization. The sampled chains contain graph-specific entity and relation identities. We therefore tokenize each chain $C_{i,j}$ into query-relative structural tokens that preserve its organization and chain order, discarding identities that do not transfer across KGs.

Entity tokenization. For an entity e in the support chain, let $d_i^h(e)$ and $d_i^t(e)$ denote its shortest-path distances to the head u_i and tail v_i. Its structural token is defined as

$$\mathbf{x}^e_{i,j,\ell} = \mathbf{E}^e\big[d_i^h(e_{i,j,\ell}), d_i^t(e_{i,j,\ell})\big]. \tag{3}$$

where $\mathbf{E}^e$ is a learnable hop-label embedding table shared across KGs. This representation depends only on the entity's structural position relative to the local graph context.

Relation tokenization. For the ℓ-th relation, we use

$$\mathbf{x}^r_{i,j,\ell} = \mathbf{e}^{\mathrm{rel}}_{\mathrm{type}} + \mathbf{e}^{\mathrm{pos}}_{\ell} + \mathbb{I}[r_{i,j,\ell} = r_q]\mathbf{e}^{\mathrm{qry}}. \tag{4}$$

where $\mathbf{e}^{\mathrm{rel}}_{\mathrm{type}}$ is a shared type embedding, $\mathbf{e}^{\mathrm{pos}}_{\ell}$ encodes chain position, and $\mathbf{e}^{\mathrm{qry}}$ marks the query relation. Importantly, relations are represented only by their structural positions in the chain.

The tokenizer therefore maps $C_{i,j}$ into an alternating token sequence

$$\mathbf{X}_{i,j} = \big[\mathbf{x}^e_{i,j,0}, \mathbf{x}^r_{i,j,1}, \mathbf{x}^e_{i,j,1}, \ldots, \mathbf{x}^r_{i,j,L}, \mathbf{x}^e_{i,j,L}\big]. \tag{5}$$

In this way, chains from different KGs are represented in a shared structural space.

3.2 Hierarchical Grammar Encoder

To abstract invariant reasoning patterns, we propose the **Hierarchical Grammar Encoder**, which builds a word–chain–grammar hierarchy over a discrete structural vocabulary. This module progressively abstracts structural chains through word-, chain-, and grammar-level representations.

Word-Level Quantization. Structural grammar requires a vocabulary of reusable units. Inspired by how words serve as basic units in language grammar, we first learn a discrete vocabulary of local structural roles from contextualized relation states.

Specifically, ChainGram contextualizes each chain as $\mathbf{Z}_{i,j} = \mathrm{Trans}_{\mathrm{ctx}}(\mathbf{X}_{i,j})$. We then quantize each relation state using a word-level codebook $\mathcal{B}^{\mathrm{Word}}$:

$$\widehat{\mathbf{z}}^r_{i,j,\ell} = \mathrm{VQ}\big(\mathrm{LN}(\mathbf{z}^r_{i,j,\ell}); \mathcal{B}^{\mathrm{Word}}\big). \tag{6}$$

ChainGram further replaces all relation tokens in $\mathbf{Z}_{i,j}$ with their codes and get $\widehat{\mathbf{Z}}_{i,j}$. In this way, relation roles are mapped to discrete codes, forming the vocabulary of our structural grammar.

Chain-Level Quantization. Word-level codes capture local roles, while transferable reasoning also depends on their chain-level composition. We therefore learn chain-level prototypes. Specifically, we prepend a learnable summary token and encode the full chain:

$$\mathbf{H}_{i,j} = \mathrm{Trans}_{\mathrm{chain}}\big([\mathbf{z}^{\mathrm{sum}}; \widehat{\mathbf{Z}}_{i,j}]\big), \qquad \mathbf{h}_{i,j} = \mathbf{H}_{i,j}[0]. \tag{7}$$

Then, ChainGram quantizes the chain representation with the chain-level codebook $\mathcal{B}^{\mathrm{Chain}}$:

$$\widehat{\mathbf{h}}_{i,j} = \mathrm{VQ}\big(\mathrm{LN}(\mathbf{h}_{i,j}); \mathcal{B}^{\mathrm{Chain}}\big). \tag{8}$$

The resulting discrete prototypes serve as chain-level units. We further propagate this global prototype back to each relation state through:

$$\widetilde{\mathbf{H}}^r_{i,j,\ell} = \mathrm{LN}\big(\mathbf{H}^r_{i,j,\ell} + \mathbf{g}_{i,j} \odot \mathbf{W}_c \widehat{\mathbf{h}}_{i,j}\big). \tag{9}$$

where $\mathbf{g}_{i,j}$ is a learnable gate guided by the chain representations. This allows each relation state to incorporate information from the complete chain while preserving its local representation.

Grammar-Level Pattern Aggregation. A query relation is typically supported by multiple chains that exhibit recurring structural configurations. We therefore aggregate the chain representations within the query into a relation-specific grammar pattern.

Specifically, for each support chain, we construct three complementary views:

$$\mathbf{z}_{i,j}^{\text{cont}} = \text{Pool}_\ell\big(\widetilde{\mathbf{H}}_{i,j,\ell}^{r}\big), \quad \mathbf{z}_{i,j}^{\text{word}} = \text{Pool}_\ell\big(\mathbf{E}^{\text{word}}[k_{i,j,\ell}^{\text{word}}]\big), \quad \mathbf{z}_{i,j}^{\text{chain}} = \mathbf{E}^{\text{chain}}[k_{i,j}^{\text{chain}}]. \tag{10}$$

where $k_{i,j,\ell}^{\text{word}}$ and $k_{i,j}^{\text{chain}}$ are code assignments, and $\mathbf{E}^{\text{word}}$ and $\mathbf{E}^{\text{chain}}$ are their learnable embedding tables. A selector combines these views to estimate the importance of each chain:

$$\eta_{i,j} = \text{softmax}_j\big(f_{\text{sel}}(\mathbf{z}_{i,j}^{\text{cont}}, \mathbf{z}_{i,j}^{\text{word}}, \mathbf{z}_{i,j}^{\text{chain}})/\tau\big). \tag{11}$$

The weighted relation states are then aggregated to their local relation slots and summarized across support chains:

$$\mathbf{h}_{r_q}^{\text{pat}} = \text{Aggregate}\big(\{\eta_{i,j}\widetilde{\mathbf{H}}_{i,j}^{r}\}_{i,j}\big). \tag{12}$$

The resulting $\mathbf{h}_{r_q}^{\text{pat}}$ captures the structural information associated with r_q and serves as the relation-specific grammar pattern for subsequent entity-side reasoning.

3.3 Grammar-Guided Entity Reasoner

Given the relation-specific grammar pattern $\mathbf{h}_{r_q}^{\text{pat}}$, the **Grammar-Guided Entity Reasoner** matches it against query-rooted chains to identify relevant evidence for entity reasoning. It consists of two stages: *GNN-Based Entity Encoding* and *Grammar-Guided Chain Matching*.

GNN-Based Entity Encoding. Following the general KGFM framework (Cui et al., 2024), we encode the observed graph with a query-conditioned GNN. Given $q = (s_q, r_q, ?)$, we initialize the relation states with the relation grammar patterns and activate only query entity:

$$\mathbf{V}_{\text{R}}^{(0)}[r] = \mathbf{h}_r^{\text{pat}}, \qquad \mathbf{V}_{\text{E}}^{(0)}[v] = \mathbb{I}[v = s_q]\mathbf{q}. \tag{13}$$

We perform query-conditioned message passing over the observed graph. For each edge (u, r, v) at layer ℓ, entity states are updated by

$$\mathbf{h}_v^{(\ell+1)} = \text{GNNUpdate}\Big(\mathbf{h}_v^{(\ell)}, \textstyle\sum_{(u,r,v)\in\mathcal{N}(v)} \alpha_{u,r,v}^{(\ell)}\big(\mathbf{h}_u^{(\ell)} + \mathbf{h}_r^{(\ell)}\big)\Big). \tag{14}$$

where $\alpha_{u,r,v}^{(\ell)}$ is an attention weight conditioned on the source entity and relations. After N rounds of graph propagation, we obtain entity representations $\mathbf{h}_v^{(N)}$ and base scores $s_{\text{base}}(v \mid q)$.

Grammar-Guided Chain Matching. The GNN provides contextualized entity representations, but different query-rooted chains may contribute unequally to the current prediction. We therefore use the learned relation-specific grammar pattern to identify the chains whose structural configurations are most relevant to the query.

Specifically, we sample N fixed-length entity–relation chains rooted at s_q and encode each chain as

$$\mathbf{g}_n = \text{Trans}_{\text{ent}}\big([\mathbf{y}_q, \mathbf{y}_{n,0}^{e}, \mathbf{y}_{n,1}^{r}, \ldots, \mathbf{y}_{n,L}^{e}]\big)[0], \tag{15}$$

where the entity tokens incorporate the GNN representations. Then, we match each chain against the grammar pattern of r_q:

$$\mathbf{a}_n = \text{MHA}\big(\mathbf{g}_n, \mathbf{h}_{r_q}^{\text{pat}}, \mathbf{h}_{r_q}^{\text{pat}}\big), \qquad m_n = f_{\text{match}}\big([\mathbf{g}_n \| \mathbf{a}_n \| \mathbf{g}_n \odot \mathbf{a}_n]\big). \tag{16}$$

The matching score m_n measures how well the structural configuration of the n-th query chain agrees with the learned grammar pattern. We normalize these scores to obtain chain relevance weights and aggregate the corresponding evidence:

$$\omega_n = \text{softmax}_{n\in\mathcal{I}_q}(m_n), \qquad \mathbf{c}_q = \textstyle\sum_{n\in\mathcal{I}_q} \omega_n \mathbf{g}_n. \tag{17}$$

This pattern-guided matching explicitly connects the abstract relation-level grammar with concrete query-side evidence. Finally, we use the aggregated chain evidence as a gated residual to refine the GNN-based entity prediction:

$$s(v \mid q) = s_{\text{base}}(v \mid q) + \gamma_q \frac{(\mathbf{W}_e \mathbf{h}_v^{G})^\top (\mathbf{W}_c \mathbf{c}_q)}{\sqrt{d}}. \tag{18}$$

where γ_q is a query-dependent gate. In this way, the final prediction combines graph-level representations with the chain evidence identified by the learned grammar pattern. Training objectives, algorithm details, and complexity analysis are provided in Appendix C.

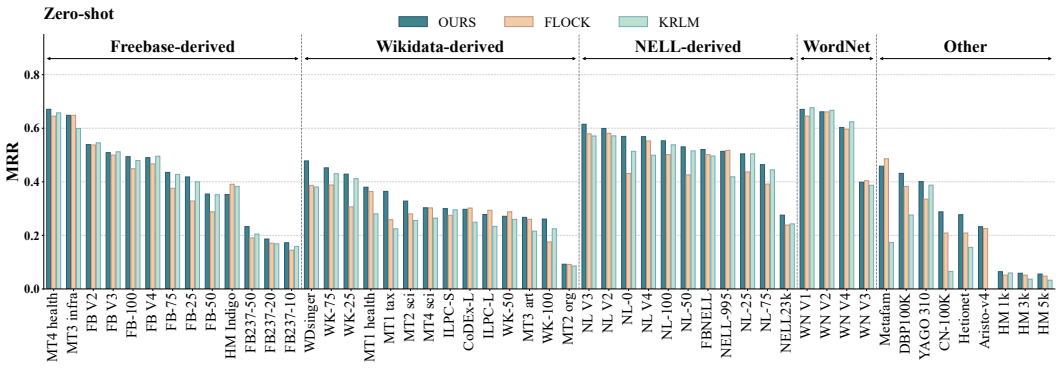


Figure 3: Zero-shot MRR (higher is better) in the entity prediction task

4 EXPERIMENTS

We evaluate CHAINGRAM through extensive experiments guided by following research questions:

- **RQ1:** How does CHAINGRAM compare with SOTA KGFMs in reasoning across KGs?
- **RQ2:** Does CHAINGRAM learn grammar patterns that transfer consistently across KGs?
- **RQ3:** How does each component of CHAINGRAM contribute to its overall effectiveness?
- **RQ4:** How do pretraining scale and knowledge sources influence the generalization ability?

4.1 EXPERIMENTAL SETUP

Datasets and Implementations. We pretrain all models on the same three source KGs used by KG-ICL (Cui et al., 2024): FB-v1, NELL-v1, and CoDEx-S. For downstream evaluation, we use 54 target KGs grouped into three settings: *Inductive-e/r* (23 graphs), where unseen entities and relation types may appear at inference time; *Inductive-e* (16 graphs), where entities are unseen while the relation vocabulary is fixed; and *Transductive* (15 graphs), where entity and relation vocabularies are shared between training and inference. We use identical dataset splits and preprocessing across all methods. Full dataset lists, statistics, and processing details are provided in Appendix D.1.1.

Baselines. We compare CHAINGRAM with state-of-the-art KGFMs spanning two major paradigms. The **structure-based** baselines include ULTRA (Galkin et al., 2024), KG-ICL (Cui et al., 2024), TRIX (Zhang et al., 2025), MOTIF (Huang et al., 2025a), and FLOCK (Kim et al., 2026). The **LLM-enhanced** baselines include Prolink (Wang et al., 2024), SEMMA (Arun et al., 2025), and KRLM (Zhuo et al., 2026). All baselines are retrained on the same source KGs and evaluated on target datasets and splits. Detailed descriptions and implementation are provided in Appendix D.1.2.

Evaluation Protocol. We evaluate all KGFMs using MRR and Hits@10 under filtered ranking protocol. Since target KGs vary substantially in size, we compute metrics on each KG and report macro-averages. We consider two protocols: *zero-shot*, which directly evaluates the pretrained model; *full fine-tuning*, which adapts the model using available target-graph training triples. Further details of query construction, ranking, and filtered ranking are provided in Appendix D.1.3.

4.2 PERFORMANCE COMPARISON (*RQ1*)

Table 1 summarizes the macro-averaged results under zero-shot and full fine-tuning protocols. Overall, CHAINGRAM consistently outperforms strong KGFM baselines across different transfer settings. Under **zero-shot** transfer, CHAINGRAM achieves the best performance on all metrics, improving overall MRR by 2.82% over the strongest baseline, with a larger gain on Inductive-E,R (+3.41%). Under **full fine-tuning**, CHAINGRAM ranks first on all metrics and improves overall MRR by 5.84%, demonstrating that the learned structural patterns remain effective after target-specific adaptation.

Figure 3 further shows zero-shot MRR on individual target KGs. The improvements of CHAINGRAM are consistently observed across different KG families, including Freebase-derived, Wikidata-derived, NELL-derived, and WordNet datasets, rather than being dominated by a small

Table 1: Macro-averaged entity prediction results under zero-shot and full fine-tuning. Existing KGFMs are grouped into structure-based and LLM-enhanced methods. Best and second-best results are shown in **bold** and underline, respectively.

Protocol	Model	Inductive-E,R (23)		Inductive-E (16)		Transductive (15)		Overall (54)	
		MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
Zero-shot	*Structure-based KGFMs*								
	ULTRA	0.3151	0.4873	0.3830	0.5251	0.2526	0.3892	0.3179	0.4712
	KG-ICL	0.4107	0.5897	0.4112	0.5437	0.3228	0.4670	0.3864	0.5420
	TRIX	0.3378	0.5153	0.4042	0.5453	0.2752	0.4127	0.3401	0.4957
	MOTIF	0.3191	0.5008	0.3856	0.5154	0.1891	0.2752	0.3027	0.4425
	FLOCK	0.3749	0.5500	0.4146	0.5544	0.2728	0.4026	0.3583	0.5104
	LLM-enhanced KGFMs								
	ProLink	0.3065	0.4759	0.3767	0.5252	0.2397	0.3769	0.3087	0.4630
	SEMMA	0.3116	0.4882	0.3693	0.5179	0.2407	0.3699	0.3090	0.4641
	KRLM	0.3808	0.5689	0.4121	0.5552	0.2635	0.4045	0.3575	0.5192
	Chain-based structural model (Ours)								
	CHAINGRAM	**0.4247**	**0.6069**	**0.4227**	**0.5562**	**0.3280**	**0.4725**	**0.3973**	**0.5546**
Full FT	*Structure-based KGFMs*								
	ULTRA	0.3981	0.5581	0.3846	0.5437	0.3566	0.5097	0.3826	0.5404
	KG-ICL	0.4340	0.5902	0.4354	0.5632	0.3676	0.5057	0.4160	0.5587
	TRIX	0.3908	0.5434	0.4152	0.5446	0.2996	0.4386	0.3727	0.5146
	MOTIF	0.2984	0.4395	0.3443	0.4672	0.2525	0.3903	0.2993	0.4340
	FLOCK	0.4057	0.5630	0.4382	0.5654	0.3515	0.5015	0.4003	0.5466
	LLM-enhanced KGFMs								
	ProLink	0.3914	0.5489	0.3909	0.5418	0.2840	0.4338	0.3614	0.5148
	SEMMA	0.3709	0.5394	0.4173	0.5512	0.3309	0.4805	0.3736	0.5265
	KRLM	0.4365	0.6206	0.4362	0.5678	0.3363	0.4959	0.4086	0.5703
	Chain-based structural model (Ours)								
	CHAINGRAM	**0.4643**	**0.6247**	**0.4423**	**0.5679**	**0.4012**	**0.5432**	**0.4403**	**0.5852**

subset of graphs. This indicates that the learned structural language provides robust transferability across heterogeneous KGs. Additional results on individual datasets and full comparison details are provided in Appendix D.2.

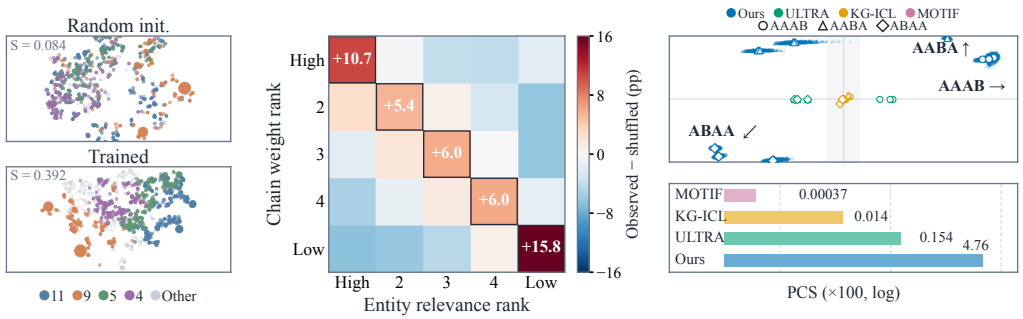
4.3 UNDERSTANDING THE LEARNED STRUCTURAL GRAMMAR (*RQ2*)

We investigate whether CHAINGRAM learns a structural grammar that captures invariant patterns across KGs. Specifically, we examine three aspects of the learned structural grammar: whether the *grammar encoder* organizes chains into structured prototypes, whether the *reasoner* identifies query-relevant chains, and whether the learned patterns remain *consistent across different KGs*. Finally, we provide a case study to illustrate the structural similarities among chains assigned to the same code across KGs. Detailed experimental protocols and additional results are provided in Appendix D.3.

(a) Prototype Separability. We examine whether training makes chain representations assigned to different codes more separable. We visualize the chain representations from the randomly initialized and after trained encoder, with colors indicating their learned chain-level code. As shown in Fig. 4(a), the trained encoder forms clearer prototype-aligned clusters, showing that CHAINGRAM learns an more separability discrete structural space.

(b) Evidence Relevance. We examine whether the grammar-guided reasoner assigns larger weights to more informative chains. For each query, we compare the learned chain weights with each chain's ability to distinguish the ground-truth entity from negatives under a within-query shuffled control. Fig. 4(b) shows the difference from this baseline; diagonal enrichment indicates agreement between chain importance and entity relevance. All diagonal cells show positive gains (5.4–15.8 percentage points), indicating that the reasoner favors chains with stronger evidence for the correct entity.

(c) Cross-KG Pattern Transferability. We further examine whether CHAINGRAM captures KG-invariant patterns beyond graph-specific identities. Since real KGs are noisy and provide no explicit



(a) Separability (b) Relevance (c) Transferability

Figure 4: (a) **Separability**: trained representations form clearer prototype regions. (b) **Relevance**: chains assigned higher weights are more likely to correspond to more relevant evidence. (c) **Transferability**: structural patterns transfer more consistently across KGs than existing implicit methods.

correspondence between structural patterns, we design a controlled synthetic probe: graph statistics are matched and relation identities are permuted, while the underlying structural patterns are preserved. We measure cross-KG consistency using pattern consistency score (PCS), which tests whether representations of the same structural pattern are closer across KGs than those of different patterns. As shown in Fig. 4(c), CHAINGRAM achieves substantially higher PCS than existing KGFMs, indicating that it better preserves structural equivalence across graphs and learns transferable structural patterns beyond graph-specific relation identities.

(d) Structural Code Case Study. We conduct a case study to examine whether chains assigned to the same code share similar structural patterns. Fig. 5 shows three Code 8 examples following the same relation pattern. In ILPC2022-small and WKIngram-50, chains composed of semantically corresponding relations receive the same code, although the graphs are processed independently. The FB15K237-10 example uses different relations but follows a similar $(r_A, r_B, r_C, r_C^{-1})$ pattern. Each chain reaches a related entity through an institution or location, then connects to another entity through a shared relation. These examples provide qualitative evidence that CHAINGRAM captures structural grammar through recurring chain patterns across KGs.

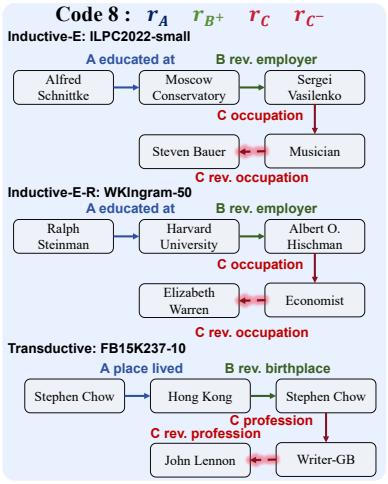


Figure 5: Case Study of chain codes.

4.4 ABLATION STUDIES (*RQ3*)

We conduct ablation experiments to examine the contribution of each component in CHAINGRAM. Specifically, we consider four variants: **M1: w/o Chain Reasoning**, which removes the explicit chain-based reasoning branch and relies only on graph-level representations; **M2: w/o Word-Level Structural Roles**, which removes the word-level codebook and replaces discrete local structural roles with continuous representations; **M3: w/o Chain-Level Prototypes**, which removes the chain-level codebook and represents chain-level compositions using continuous representations; and **M4: w/o Grammar-Guided Reasoner**, which removes the grammar-guided chain matching module and relies on the base GNN prediction.

The results are shown in Fig. 6. Overall, the complete CHAINGRAM achieves the strongest performance across different transfer settings, while removing any major component leads to consistent degradation. First, **M1** causes the largest performance drop, highlighting the importance of explicit chain reasoning beyond graph-level message passing. **M2** further demonstrates the necessity of query-relative structural roles, as replacing discrete structural abstractions with continuous representations weakens cross-KG transfer. **M3** and **M4** show that both hierarchical pattern abstraction and grammar-guided chain matching provide complementary benefits beyond continuous chain modeling. Overall, these results validate the effectiveness of organizing transferable reasoning through

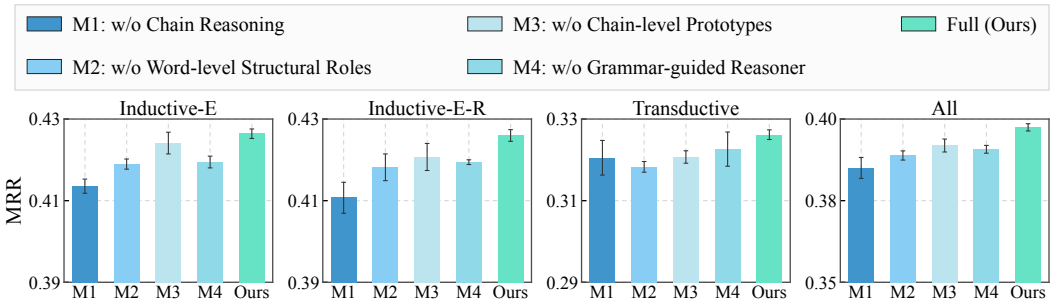


Figure 6: Ablation study results, showing performance impact of removing different components.

structural patterns and grounding them with query-specific evidence. Additional ablation results are provided in Appendix D.4.

4.5 Analysis of Pretraining and Knowledge Sources (*RQ4*)

We investigate how pretraining data and knowledge sources affect transferability. We study three aspects: (i) robustness to different pretraining KGs, (ii) the impact of pretraining scale, (iii) comparison with LLM-enhanced methods for KG reasoning.

(a) Pretraining KG Distribution. We first examine whether ChainGram's effectiveness depends on the choice of source KGs. Replacing our source KGs with the standard three-KG pretraining configuration used by prior KGFMs (Galkin et al., 2024; Kim et al., 2026), ChainGram still achieves competitive zero-shot performance, with an overall MRR 1.61% improvement over the strongest baseline. Full results are provided in Appendix D.5.1

(b) Pretraining Scale. We further study whether more diverse pretraining KGs improve transferability. We vary the number of source KGs from one to seven and evaluate on the same 27 unseen target KGs. As shown in Fig. 7, performance improves consistently as more source KGs are added, with relative gains of 10.2% in macro MRR and 8.7% in Hits@10. This indicates that exposure to more diverse structural patterns helps ChainGram learn a broader structural grammar. Comprehensive experiment results can be found in Appendix D.5.2

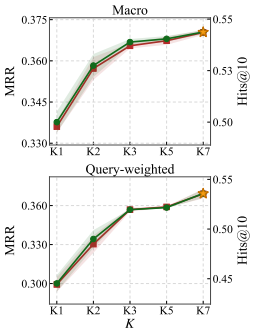


Figure 7: Effect of expanding KGs.

(c) Comparison with LLM-Enhanced KGFMs. Finally, we compare ChainGram with LLM-enhanced KGFMs to examine the role of textual knowledge in KG transfer. Since textual descriptions are often incomplete or unavailable in real-world KGs, we first divide test KGs into text-available and text-missing groups. LLM-enhanced methods exhibit an average MRR degradation of 43.0% on text-missing KGs. To further isolate the effect of textual information, we remove entity descriptions from originally text-rich KGs while keeping graph structures unchanged. Under this controlled setting, LLM-enhanced methods suffer an MRR decrease of 3.25% points, whereas ChainGram remains unchanged. These results suggest that structural patterns provide a complementary source of transferable knowledge when textual information is incomplete. Detailed results are provided in Appendix D.5.3.

5 Conclusion

In this work, we introduce ChainGram, a chain-based knowledge graph foundation model that learns and applies a transferable structural language. Through hierarchical codebooks and support-chain aggregation, ChainGram organizes structural information into word-level codes, chain-level prototypes, and relation-specific grammar patterns. These patterns guide query-rooted chain matching, connecting abstract structural knowledge with concrete evidence for entity prediction. Experiments on 54 target KGs demonstrate improved overall performance over strong baselines under zero-shot and full fine-tuning protocols. These results highlight the value of explicitly learning and reusing structural language for transferable knowledge graph reasoning.

AI USE STATEMENT

We have not used generative AI tools for generating synthetic datasets or implementing the proposed model, and tasks involving mathematical proof development or writing, hypothesis formulation or refinement, dataset cleaning and reformatting, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for language polishing and creating illustrative icons for figures. We have reviewed all AI-assisted work. Specifically, we checked that language revisions preserved the intended technical meaning and that the generated icons were consistent with the concepts illustrated in the figures. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Kossi Amouzouvi, Robert Wardenga, Jens Lehmann, and Sahar Vahdati. Graphlets as building blocks for structural vocabulary in knowledge graph foundation models. *arXiv preprint arXiv:2605.06154*, 2026.

Arvindh Arun, Sumit Kumar, Mojtaba Nayyeri, Bo Xiong, Ponnurangam Kumaraguru, Antonio Vergari, and Steffen Staab. SEMMA: A semantic aware knowledge graph foundation model. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31825–31848, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1621. URL https://aclanthology.org/2025.emnlp-main.1621/.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. In *Proceedings of the 3rd Conference on Automated Knowledge Base Construction*, 2021. URL https://openreview.net/forum?id=Qa3uS3H7-Le.

Yuanning Cui, Zequn Sun, and Wei Hu. A prompt-based knowledge graph foundation model for universal in-context reasoning. In *Advances in Neural Information Processing Systems*, volume 37, 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 *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Syg-YfWCW.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Boyang Ding, Quan Wang, Bin Wang, and Li Guo. Improving knowledge graph embedding using simple constraints. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 110–121. Association for Computational Linguistics, 2018.

Keyan Ding, Zhihui Zhu, Yuqi Tang, Kehua Feng, Xiang Zhuang, Hongwei Wang, Yi Yang, Huifang Du, Zhangkai Ni, Shiqi Wang, et al. Bridging data and discovery: a survey on knowledge graphs in ai for science. *National Science Review*, 13(8):nwag140, 2026.

Mikhail Galkin, Max Berrendorf, and Charles Tapley Hoyt. An open challenge for inductive link prediction on knowledge graphs. *arXiv preprint arXiv:2203.01520*, 2022.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *The Twelfth International Conference on Learning Representations*, 2024.

Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 1802–1808. AAAI Press, 2017. ISBN 9780999241103.

Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L. Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E. Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife*, 6:e26726, 2017.

Kexin Huang, Payal Chandak, Qianwen Wang, Shreyas Havaldar, Akhil Vaid, Jure Leskovec, Girish N Nadkarni, Benjamin S Glicksberg, Nils Gehlenborg, and Marinka Zitnik. A foundation model for clinician-centered drug repurposing. *Nature Medicine*, 30(12):3601–3613, 2024.

Xingyue Huang, Pablo Barcelo, Michael M. Bronstein, Ismail Ilkan Ceylan, Mikhail Galkin, Juan L Reutter, and Miguel Romero Orth. How expressive are knowledge graph foundation models? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 25021–25058. PMLR, 2025a.

Yanwen Huang, Yao Liu, Qiao Liu, Rui Hou, and Tingting Dai. Sce: Semantic consistency enhanced reinforcement learning for multi-hop knowledge graph reasoning. In *EMNLP (Findings)*, pp. 5413–5425, 2025b.

Jinwoo Kim, Xingyue Huang, Krzysztof Olejniczak, Kyungbin Min, Michael Bronstein, Seunghoon Hong, and Ismail Ilkan Ceylan. Flock: A knowledge graph foundation model via learning on random walks. In *The Fourteenth International Conference on Learning Representations*, 2026.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pp. 2873–2882. PMLR, 2018.

Ni Lao, Tom Mitchell, and William W. Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 529–539. Association for Computational Linguistics, 2011. URL https://aclanthology.org/D11-1049/.

Jaejun Lee, Chanyoung Chung, and Joyce Jiyoung Whang. Ingram: Inductive knowledge graph embedding via relation graphs. In *International conference on machine learning*, pp. 18796–18809. PMLR, 2023.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3243–3253. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1362. URL https://aclanthology.org/D18-1362/.

Shuwen Liu, Bernardo Grau, Ian Horrocks, and Egor Kostylev. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. *Advances in Neural Information Processing Systems*, 34:2034–2045, 2021.

Vanessa Lopez, Lam Hoang, Marcos Martinez-Galindo, Raúl Fernández-Díaz, Marco Luca Sbodio, Rodrigo Ordonez-Hurtado, Mykhaylo Zayats, Natasha Mulligan, and Joao Bettencourt-Silva. Enhancing foundation models for scientific discovery via multimodal knowledge graph representations. *Journal of Web Semantics*, 84:100845, 2025.

Xin Lv, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Wei Zhang, Yichi Zhang, Hao Kong, and Suhui Wu. Dynamic anticipation and completion for multi-hop reasoning over sparse knowledge graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 5694–5703. Association for Computational Linguistics, 2020.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th Biennial Conference on Innovative Data Systems Research*, 2014.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 2925–2933, 2020.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Tara Safavi and Danai Koutra. Codex: A comprehensive knowledge graph completion benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 8328–8350. Association for Computational Linguistics, 2020.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.

Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International conference on machine learning*, pp. 9448–9457. PMLR, 2020.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pp. 2071–2080. PMLR, 2016.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P Talukdar. Composition-based multi-relational graph convolutional networks. In *ICLR*, 2020.

Kai Wang, Yuwei Xu, Zhiyong Wu, and Siqiang Luo. LLM as prompter: Low-resource inductive reasoning on arbitrary knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3742–3759. Association for Computational Linguistics, 2024.

Zehong Wang, Zheyuan Liu, Tianyi Ma, Jiazheng Li, Zheyuan Zhang, Xingbo Fu, Yiyang Li, Zhengqing Yuan, Wei Song, Yijun Ma, et al. Graph foundation models: A comprehensive survey. *arXiv preprint arXiv:2505.15116*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 564–573, 2017.

Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, May 2015. URL https://www.microsoft.com/en-us/research/publication/embedding-entities-and-relations-for-learning-and-inference-in-knowledge-bases/.

Yongqi Zhang and Quanming Yao. Knowledge graph reasoning with relational digraph. In *Proceedings of the ACM web conference 2022*, pp. 912–924, 2022.

Yucheng Zhang, Beatrice Bevilacqua, Mikhail Galkin, and Bruno Ribeiro. Trix: A more expressive model for zero-shot domain transfer in knowledge graphs. In *Learning on Graphs Conference*, pp. 12:1–12:28. PMLR, 2025.

Ze Zhao, Bin Lu, Xiaoying Gan, Gu Tang, Luoyi Fu, and Xinbing Wang. Chainsformer: Numerical reasoning on knowledge graphs from a chain perspective. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pp. 2963–2976. IEEE, 2025.

Jincheng Zhou, Beatrice Bevilacqua, and Bruno Ribeiro. An ood multi-task perspective for link prediction with new relation types and nodes. *arXiv preprint arXiv:2307.06046*, 2023.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in neural information processing systems*, 34:29476–29490, 2021.

Zhaocheng Zhu, Xinyu Yuan, Michael Galkin, Louis-Pascal Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. A* net: A scalable path-based reasoning approach for knowledge graphs. *Advances in neural information processing systems*, 36:59323–59336, 2023.

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 *The Fourteenth International Conference on Learning Representations*, 2026.

A RELATED WORK

Knowledge Graph Reasoning. Knowledge graph reasoning aims to infer missing facts from observed triples. Early methods mainly learn entity and relation embeddings in a transductive setting, where all entities and relations are observed during training. Representative embedding methods include translational, bilinear, and geometric models, such as TransE, DistMult, ComplEx, and RotatE (Bordes et al., 2013; Yang et al., 2015; Trouillon et al., 2016; Sun et al., 2019). Graph neural networks further improve relational modeling by propagating messages over multi-relational structures, as in R-GCN and CompGCN (Schlichtkrull et al., 2018; Vashishth et al., 2020). Despite their effectiveness, these methods typically rely on graph-specific embeddings or a fixed vocabulary, making it difficult to generalize to unseen entities, unseen relations, or new graphs.

Inductive KG reasoning addresses this limitation by learning structural reasoning patterns instead of graph-specific entity embeddings. Subgraph-based methods such as GraIL extract enclosing subgraphs around query triples and reason from local structural patterns (Teru et al., 2020). Query-conditioned message-passing methods such as NBFNet, RED-GNN, and A*Net propagate signals over the graph to capture multi-hop evidence (Zhu et al., 2021; Zhang & Yao, 2022; Zhu et al., 2023). Fully inductive methods further extend this setting to unseen relation types by modeling relational dependencies or relation-aware messages (Lee et al., 2023). These methods significantly improve inductive generalization, but are still often designed for specific datasets.

Knowledge Graph Foundation Models. Knowledge graph foundation models (KGFMs) aim to learn transferable reasoning ability over arbitrary entities, relations, and graphs. Existing KGFMs mainly follow two routes: relation-graph-based modeling and query-conditioned context modeling. The first route abstracts domain-invariant relation dependencies through auxiliary relation graphs, relation interactions, triplet-level representations, or relational motifs, as in ULTRA, TRIX, and MOTIF (Galkin et al., 2024; Zhang et al., 2025; Huang et al., 2025a). The second route constructs structural contexts around each query, such as prompt graphs or anonymized random walks, and encodes them with message-passing or sequence models, as in KG-ICL and FLOCK (Cui et al., 2024; Kim et al., 2026). Recent studies further explore explicit structural vocabularies or semantic grounding, such as graphlet-based structural tokens and semantic-aware relation modeling (Amouzouvi et al., 2026; Arun et al., 2025). Despite these advances, relation-chain reasoning patterns are still mostly learned implicitly: relation dependencies are abstracted at the graph or motif level, while chain-like evidence is absorbed into broader prompt-graph, subgraph, or walk representations. Our work instead treats relation-chain patterns as explicit structural units, and further tokenizes and selects ordered relation compositions for KGFM reasoning.

Path-based Knowledge Graph Reasoning. Knowledge graph reasoning often requires evidence beyond a single observed triple, as answering a query may depend on multiple relational hops that collectively form a reasoning path. Early path-based methods such as PRA (Lao et al., 2011) exploit such multi-hop evidence through constrained random walks to discover predictive relational paths between entities. Reinforcement-learning-based approaches further formulate path discovery as a sequential decision process: MINERVA (Das et al., 2018) learns a query-conditioned policy to navigate from a source entity to potential answers, while MultiHopKG (Lin et al., 2018) introduces reward shaping to improve multi-hop exploration under sparse rewards and incomplete knowledge graphs. More recent studies have focused on improving the scalability and reliability of path-based reasoning. A*Net (Zhu et al., 2023) learns an A*-style priority function to selectively expand informative nodes and edges, substantially reducing the search space for large-scale knowledge graph reasoning. SCE (Huang et al., 2025b) further addresses the issue of spurious reasoning paths in reinforcement-learning-based methods by incorporating semantic consistency between intermediate reasoning steps and the target triple into the reward function. Beyond conventional knowledge graph completion, relational-path reasoning has also been extended to richer reasoning tasks, such as numerical reasoning with relational and attribute chains in ChainsFormer (Zhao et al., 2025). However, existing path-based approaches primarily focus on discovering individual reasoning paths within a KG, rather than learning transferable structures shared across different KGs. This motivates us to explore relational chains as reusable structural units for KGFM reasoning.

B Additional Preliminaries

B.1 Knowledge Graph Reasoning

A knowledge graph (KG) represents relational knowledge as a set of entities and relations. Formally, a KG is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$, where $\mathcal{V}$ denotes the entity set, $\mathcal{R}$ denotes the relation set, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ denotes the observed facts. Each fact is represented as a triple (h, r, t), where $h, t \in \mathcal{V}$ and $r \in \mathcal{R}$.

The goal of KG reasoning is to infer missing facts from observed graph structures. Given a query triple with one missing entity, $(h, r, ?)$ or $(?, r, t)$, the model ranks candidate entities according to the likelihood of forming a valid triple. Following standard practice in link prediction, we introduce an inverse relation r^{-1} for each relation r, allowing head prediction queries to be converted into tail prediction queries:

$$(?, r, t) \rightarrow (t, r^{-1}, ?). \tag{19}$$

Therefore, all queries can be uniformly represented as $q = (s_q, r_q, ?)$.

Given a query q and an observed graph context $\mathcal{G}_{\text{inf}}$, a reasoning model with parameters θ produces a score for each candidate entity:

$$s(v|q) = s_\theta(\mathcal{G}_{\text{inf}}, s_q, r_q, v), \tag{20}$$

where higher scores indicate that the candidate entity v is more likely to complete the query triple.

B.2 Knowledge Graph Foundation Models

Knowledge graph foundation models (KGFMs) aim to learn transferable reasoning capabilities from multiple source knowledge graphs and apply them to unseen target graphs (Wang et al., 2025). Unlike conventional KG reasoning models that are trained and evaluated on the same graph, KGFMs focus on cross-graph generalization, where the target KG may differ from source KGs in entities, relations, and graph structures.

Formally, a KGFM is pretrained on a collection of source KGs:

$$\mathbb{G}_{\text{src}} = \{\mathcal{G}_1, \ldots, \mathcal{G}_S\}, \tag{21}$$

where each source graph provides structural contexts for learning transferable reasoning patterns. After pretraining, the model is evaluated on target KGs that are not involved during pretraining:

$$\mathbb{G}_{\text{tar}} = \{\mathcal{G}'_1, \ldots, \mathcal{G}'_T\}. \tag{22}$$

For each target KG, evaluation is performed based on the observed graph context available at inference time, $\mathcal{G}_{\text{inf}}$.

B.3 Transfer Evaluation Settings

KGFM evaluation focuses on whether reasoning patterns learned from source KGs can generalize to unseen target KGs. Given a target KG $\mathcal{G}_{\text{tar}} = (\mathcal{V}_{\text{tar}}, \mathcal{R}_{\text{tar}}, \mathcal{E}_{\text{tar}})$, the model performs prediction based on an observed graph context $\mathcal{G}_{\text{inf}} \subseteq \mathcal{G}_{\text{tar}}$ available at inference time. The remaining facts are used as prediction targets according to the evaluation protocol.

Following existing KGFM studies (Galkin et al., 2024; Cui et al., 2024), we categorize transfer scenarios according to the relationship between the observed graph and the inference graph vocabularies.

In the *transductive* setting, entity and relation vocabularies are shared between the observed and inference graphs:

$$\mathcal{V}_{\text{inf}} = \mathcal{V}_{\text{tar}}, \qquad \mathcal{R}_{\text{inf}} = \mathcal{R}_{\text{tar}}. \tag{23}$$

The model reasons over entities and relations that have been observed in the target KG.

In the *inductive-e* setting, inference involves unseen entities while relation types remain shared:

$$\mathcal{V}_{\text{inf}} \neq \mathcal{V}_{\text{tar}}, \qquad \mathcal{R}_{\text{inf}} = \mathcal{R}_{\text{tar}}. \tag{24}$$

This setting evaluates whether learned structural patterns can transfer to entities that are not observed in the available graph context.

In the *Inductive-E,R* setting, both unseen entities and unseen relation types may appear:

$$\mathcal{V}_{\text{inf}} \neq \mathcal{V}_{\text{tar}}, \qquad \mathcal{R}_{\text{inf}} \neq \mathcal{R}_{\text{tar}}. \tag{25}$$

This represents the most challenging scenario, requiring transfer beyond both entity and relation vocabularies.

B.4 Vector Quantization and Codebook Learning

To learn a discrete structural vocabulary, ChainGram adopts vector quantization (VQ) to map continuous representations into a finite set of learnable codebook entries. Given a codebook

$$\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_K\},$$

where each $\mathbf{b}_k \in \mathbb{R}^d$ represents a learnable structural prototype, VQ assigns an input representation $\mathbf{x} \in \mathbb{R}^d$ to its nearest code vector:

$$k^* = \arg\min_k \|\mathbf{x} - \mathbf{b}_k\|_2^2. \tag{26}$$

The quantized representation is then obtained by replacing the continuous vector with the selected codebook entry:

$$\mathbf{z} = \mathbf{b}_{k^*}. \tag{27}$$

Since the nearest-neighbor assignment is non-differentiable, we adopt the straight-through estimator used in VQ-VAE (Van Den Oord et al., 2017):

$$\text{VQ}(\mathbf{x}; \mathcal{B}) = \mathbf{x} + \text{sg}[\mathbf{b}_{k^*} - \mathbf{x}], \tag{28}$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operator. During the forward pass, the output is equivalent to the selected codebook entry, while gradients are propagated through the encoder representation.

Following the standard VQ objective, the codebook and encoder are optimized jointly with commitment losses:

$$\mathcal{L}_{\text{vq}} = \|\,\text{sg}[\mathbf{x}] - \mathbf{b}_{k^*}\|_2^2 + \beta\|\mathbf{x} - \text{sg}[\mathbf{b}_{k^*}]\|_2^2, \tag{29}$$

where the first term updates the codebook entries and the second term encourages encoder outputs to commit to selected codes.

In ChainGram, this mechanism is applied hierarchically with separate codebooks. The word-level codebook captures reusable local structural roles, while the chain-level codebook captures recurring chain-level compositions. These discrete codes provide the basis for learning a structural vocabulary and its higher-level grammar.

Existing KGFMs have explored different structural abstractions for cross-KG reasoning, including relation dependencies, graph motifs, and query-conditioned contexts. These abstractions demonstrate that transferable structural information exists across KGs, but differ in the granularity and organization of the learned structures.

Table C.1: Comparison of structural abstraction paradigms in representative KGFM approaches.

Method	Structural Unit	Representation	Ordered Composition
ULTRA	Relation graph	Relation dependencies	Implicit
KG-ICL	Prompt graph	Context representation	Implicit
TRIX	Relation structure	Structural relation representation	Partial
MOTIF	Motifs	Structural motifs	Implicit
FLOCK	Random walks	Walk representations	Sequential
CHAINGRAM	**Relational chains**	**Hierarchical structural grammar**	**Explicit**

C Additional Method Details

C.1 Structural Grammar Design and KGFM Abstraction Comparison

Existing knowledge graph foundation models (KGFMs) have demonstrated that invariant structural patterns exist across knowledge graphs, but different methods abstract such patterns at different levels. Representative approaches model relation dependencies through auxiliary relation graphs, capture local topology through motifs, or encode query-specific contexts through subgraphs and random walks. These abstractions provide effective structural representations for cross-KG reasoning, but they generally summarize relational information at the level of dependencies or contexts, where the compositional organization of multi-hop relations remains implicit.

CHAINGRAM focuses on a different abstraction level by treating relational chains as explicit structural units. A relational chain preserves the ordered composition of multiple relational steps:

$$r_1 \to r_2 \to \cdots \to r_L,$$

which provides a natural basis for capturing reusable structural compositions across KGs. Instead of directly encoding chains into continuous representations, CHAINGRAM organizes these compositions into a hierarchical structural grammar consisting of local relation roles, chain-level prototypes, and relation-specific grammar patterns.

Table C.1 summarizes the differences between CHAINGRAM and representative KGFM paradigms from the perspective of structural abstraction.

C.2 Learning Objective

We train ChainGram end-to-end by jointly optimizing the entity-ranking objective and the two hierarchical vector-quantization objectives.

Entity-Ranking Objective. For a query $q = (s_q, r_q, ?)$ with ground-truth answer o_q, the Grammar-Guided Entity Reasoner produces a score $s(v \mid q)$ for each candidate entity $v \in \mathcal{V}_q$. We convert these scores into a probability distribution over candidate entities:

$$p(v \mid q) = \frac{\exp(s(v \mid q))}{\sum_{u \in \mathcal{V}_q} \exp(s(u \mid q))}. \tag{30}$$

The entity-ranking loss is defined as the negative log-likelihood of the correct answer:

$$\mathcal{L}_{\text{rank}} = -\frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \log p(o_q \mid q), \tag{31}$$

where $\mathcal{Q}$ denotes the set of training queries in a mini-batch.

Hierarchical Quantization Objective. The Hierarchical Grammar Encoder performs vector quantization at both the word and chain levels. Following the VQ objective defined in Appendix B.4, the word-level quantization loss is computed over the contextualized relation states:

$$\mathcal{L}_{\text{vq}}^{\text{word}} = \frac{1}{|\Omega_{\text{word}}|} \sum_{(i,j,\ell) \in \Omega_{\text{word}}} \mathcal{L}_{\text{vq}} \left(\text{LN}(\mathbf{z}_{i,j,\ell}^{r}); \mathcal{B}^{\text{word}} \right), \tag{32}$$

where Ω_{word} contains all valid relation positions in the sampled support chains.

Similarly, the chain-level objective is computed over the complete chain representations:

$$\mathcal{L}_{\text{vq}}^{\text{chain}} = \frac{1}{|\Omega_{\text{chain}}|} \sum_{(i,j)\in\Omega_{\text{chain}}} \mathcal{L}_{\text{vq}} \left(\text{LN}(\mathbf{h}_{i,j}); \mathcal{B}^{\text{chain}} \right), \tag{33}$$

where Ω_{chain} denotes the set of valid sampled support chains.

Overall Objective. The final training objective jointly optimizes entity prediction and hierarchical codebook learning:

$$\mathcal{L} = \mathcal{L}_{\text{rank}} + \lambda_{\text{vq}} \left(\mathcal{L}_{\text{vq}}^{\text{word}} + \mathcal{L}_{\text{vq}}^{\text{chain}} \right), \tag{34}$$

where λ_{vq} controls the contribution of the quantization objectives.

C.3 Algorithm Flow and Pseudocode

Algorithm 1 summarizes the training procedure of ChainGram. For clarity, N_s denotes the number of support chains per prompt graph, N_q the number of query-rooted chains, and T_G the number of GNN propagation layers.

For each query $q = (s_q, r_q, ?)$, the procedure comprises three stages. *First, structural chain tokenization* samples M support facts of r_q, extracts N_s chains from each support-specific prompt graph, and maps them into query-relative token sequences.

Second, hierarchical grammar encoding quantizes contextualized relation states and complete chain representations using word- and chain-level codebooks, respectively. A selector assigns chain weights normalized within each prompt graph, and the weighted relation states are aggregated by relation slot to form the pattern table $\{\mathbf{h}_r^{\text{pat}}\}_r$.

Third, grammar-guided entity reasoning uses this table to initialize the relation states for T_G layers of query-conditioned GNN propagation. The query-relation pattern $\mathbf{h}_{r_q}^{\text{pat}}$ then guides the matching and weighting of N_q query-rooted chains. Their aggregated evidence refines the base GNN prediction through a gated residual. The model is optimized jointly with the entity-ranking and two-level VQ objectives.

At inference time, ChainGram follows the same three forward stages while omitting the loss computation and parameter update. The learned parameters and codebooks are shared across KGs; no graph-specific entity or relation identity embedding is required.

C.4 Complexity Analysis

We analyze the computational complexity of ChainGram during training. Let B denote the batch size, M the number of support facts per query, N_s the number of sampled support chains per support fact, N_q the number of query-rooted chains, L the chain length, d the hidden dimension, and T_G the number of GNN propagation layers.

Chain-based Grammar Learning. For each query, ChainGram processes MN_s support chains and N_q query-rooted chains. Since each chain has a fixed length L, the Transformer encoders introduce

$$\mathcal{O}\left((MN_s + N_q)L^2 d\right) \tag{35}$$

complexity, where the quadratic term comes from self-attention over chain tokens. The hierarchical quantization and chain aggregation operations are linear with respect to the number of sampled chains and are dominated by the encoding cost.

Graph-based Entity Reasoning. The query-conditioned GNN performs sparse message passing on the observed graph. With $|\mathcal{E}_q|$ edges and $|\mathcal{V}_q|$ entities involved in reasoning, T_G propagation layers require

$$\mathcal{O}\left(T_G(|\mathcal{E}_q| + |\mathcal{V}_q|)d^2\right). \tag{36}$$

The grammar-guided chain matching over N_q query chains introduces an additional $\mathcal{O}(N_q d^2)$ cost, which is negligible compared with graph propagation and chain encoding.

Algorithm 1 Training of CHAINGRAM

Require: Observed KG $\mathcal{G}$; labeled queries $\mathcal{Q}_{\mathrm{tr}}$; learnable parameters θ, including $\mathcal{B}^{\mathrm{word}}$ and $\mathcal{B}^{\mathrm{chain}}$
Require: Support size M; neighborhood radius k; chain counts N_s, N_q; chain length L; GNN depth T_G; temperature τ; VQ weight λ_{vq}

for each minibatch $\mathcal{Q}_b \subseteq \mathcal{Q}_{\mathrm{tr}}$ **do**
  for each $(q, o_q) \in \mathcal{Q}_b$, where $q = (s_q, r_q, ?)$ **do**
  (1) Structural Chain Tokenizer
    Sample M support facts for r_q and construct their k-hop prompt graphs
    Sample N_s length-L chains per prompt graph and tokenize them into $\{\mathbf{X}_{i,j}\}$
  (2) Hierarchical Grammar Encoder
    $\mathbf{Z}_{i,j} \leftarrow \mathrm{Trans}_{\mathrm{ctx}}(\mathbf{X}_{i,j})$
    $\widehat{\mathbf{z}}^{r}_{i,j,\ell} \leftarrow \mathrm{VQ}\big(\mathrm{LN}(\mathbf{z}^{r}_{i,j,\ell}); \mathcal{B}^{\mathrm{word}}\big)$
    Replace relation states in $\mathbf{Z}_{i,j}$ with their quantized vectors to obtain $\widehat{\mathbf{Z}}_{i,j}$
    $\mathbf{H}_{i,j} \leftarrow \mathrm{Trans}_{\mathrm{chain}}\big([\mathbf{z}^{\mathrm{sum}}; \widehat{\mathbf{Z}}_{i,j}]\big)$; $\mathbf{h}_{i,j} \leftarrow \mathbf{H}_{i,j}[0]$
    $\widehat{\mathbf{h}}_{i,j} \leftarrow \mathrm{VQ}\big(\mathrm{LN}(\mathbf{h}_{i,j}); \mathcal{B}^{\mathrm{chain}}\big)$
    Apply gated prototype injection to obtain $\widetilde{\mathbf{H}}^{r}_{i,j}$
    Construct the continuous, word-level, and chain-level views; record the VQ-loss terms
    $\eta_{i,j} \leftarrow \mathrm{softmax}_j\big(f_{\mathrm{sel}}(\mathbf{z}^{\mathrm{cont}}_{i,j}, \mathbf{z}^{\mathrm{word}}_{i,j}, \mathbf{z}^{\mathrm{chain}}_{i,j})/\tau\big)$ ▷ within prompt graph i
    Aggregate $\{\eta_{i,j}\widetilde{\mathbf{H}}^{r}_{i,j}\}_{i,j}$ by relation slot to obtain $\{\mathbf{h}^{\mathrm{pat}}_{r}\}_{r}$
  (3) Grammar-Guided Entity Reasoner
    $\mathbf{V}^{(0)}_{\mathrm{R}}[r] \leftarrow \mathbf{h}^{\mathrm{pat}}_{r}$; $\mathbf{V}^{(0)}_{\mathrm{E}}[v] \leftarrow \mathbb{I}[v = s_q]\mathbf{q}$
    Run T_G query-conditioned GNN layers to obtain $\mathbf{h}^{G}_{v}$ and $s_{\mathrm{base}}(v \mid q)$
    Sample N_q length-L query-rooted chains; let $\mathcal{I}_q$ index the valid chains
    Encode each valid chain with $\mathrm{Trans}_{\mathrm{ent}}$ to obtain $\mathbf{g}_n$
    $\mathbf{a}_n \leftarrow \mathrm{MHA}\big(\mathbf{g}_n, \mathbf{h}^{\mathrm{pat}}_{r_q}, \mathbf{h}^{\mathrm{pat}}_{r_q}\big)$
    $m_n \leftarrow f_{\mathrm{match}}\big([\mathbf{g}_n \| \mathbf{a}_n \| \mathbf{g}_n \odot \mathbf{a}_n]\big)$
    $\omega_n \leftarrow \mathrm{softmax}_{n \in \mathcal{I}_q}(m_n)$; $\mathbf{c}_q \leftarrow \sum_{n \in \mathcal{I}_q} \omega_n \mathbf{g}_n$
    Compute γ_q from $\mathbf{h}^{\mathrm{pat}}_{r_q}$ and $\mathbf{c}_q$
    $s(v \mid q) \leftarrow s_{\mathrm{base}}(v \mid q) + \gamma_q d^{-1/2}(\mathbf{W}_e \mathbf{h}^{G}_{v})^{\top}(\mathbf{W}_c \mathbf{c}_q)$
    Compute $\mathcal{L}_{\mathrm{rank},q}$ using the candidate scores and answer o_q
  end for
  Average ranking losses over $\mathcal{Q}_b$ and VQ losses over valid relation positions and chains
  $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{rank}} + \lambda_{\mathrm{vq}}\big(\mathcal{L}^{\mathrm{word}}_{\mathrm{vq}} + \mathcal{L}^{\mathrm{chain}}_{\mathrm{vq}}\big)$
  Update θ by backpropagating $\mathcal{L}$
end for

Overall Complexity. Combining the above components, the per-batch complexity of CHAINGRAM is

$$\mathcal{O}\Big(B\big[(MN_s + N_q)L^2 d + T_G(|\mathcal{E}_q| + |\mathcal{V}_q|)d^2 + N_q d^2\big]\Big). \tag{37}$$

In practice, M, N_s, N_q, L, and the codebook sizes are fixed hyperparameters. Therefore, the chain-based computation scales linearly with the number of sampled chains, while graph reasoning scales with the size of the observed query graph. The attention cost depends on the chain length rather than the size of the entire KG.

D ADDITIONAL EXPERIMENTS

D.1 DETAILED SETUPS

D.1.1 DATASETS

We provide additional details on the knowledge graphs used for pretraining and downstream evaluation. In total, our experiments involve 57 KGs, consisting of 3 source KGs for pretraining and 54 target KGs for evaluation. The target KGs span diverse domains, graph scales, and generalization regimes, and are grouped into three settings: *Inductive-E,R* (23 graphs), *Inductive-e* (16 graphs),

and *Transductive* (15 graphs). Unless otherwise specified, we follow the original dataset splits and apply the same preprocessing pipeline to all evaluated methods.

Pretraining datasets. We pretrain all models on the same three source KGs used by KG-ICL (Cui et al., 2024). The pretraining corpus contains two *Inductive-e* graphs, FB15k-237 Inductive v1 and NELL Inductive v1 from the GraIL benchmark (Teru et al., 2020), and one *Transductive* graph, CoDEx-S (Safavi & Koutra, 2020). The two GraIL graphs contain unseen entities at inference time while sharing the relation vocabulary between training and inference, whereas CoDEx-S follows the standard transductive setting with a fixed entity and relation vocabulary across training and inference. Together, these source KGs expose the models to both inductive and transductive relational structures during pretraining. Detailed statistics are reported in Table D.2.

Table D.2: Statistics of the three source KGs used for pretraining.

Dataset	**Rel.**	**Training Split**		**Validation Split**			**Test Split**		
		Ent.	**Triples**	**Ent.**	**Triples**	**Valid**	**Ent.**	**Triples**	**Test**
FB15k-237 Inductive v1 (Teru et al., 2020)	180	1,594	4,245	1,594	4,245	489	1,093	1,993	411
NELL Inductive v1 (Teru et al., 2020)	14	3,103	4,687	3,103	4,687	414	225	833	201
CoDEx-S (Safavi & Koutra, 2020)	42	2,034	32,888	2,034	32,888	1,827	2,034	32,888	1,828

Inductive-E,R target datasets. The *Inductive-E,R* setting contains 23 target graphs in which both unseen entities and unseen relation types may appear at inference time. It consists of 13 InGram datasets (Lee et al., 2023) and 10 datasets adopted from MTDEA (Zhou et al., 2023). The InGram datasets cover FB-, WK-, and NL-based graph families, while the MTDEA collection includes WikiTopics graphs as well as Metafam and FBNELL. Their statistics are reported in Table D.3.

Table D.3: Target KGs and their statistics in the *Inductive-E,R* setting.

Dataset	**Training Graph**			**Test Graph**			
	Ent.	**Rel.**	**Triples**	**Ent.**	**Rel.**	**Graph**	**Test**
InGram – FB (Lee et al., 2023)							
FBIngram-25	5,190	163	91,571	4,097	216	17,147	5,716
FBIngram-50	5,190	153	85,375	4,445	205	11,636	3,879
FBIngram-75	4,659	134	62,809	2,792	186	9,316	3,106
FBIngram-100	4,659	134	62,809	2,624	77	6,987	2,329
InGram – WK (Lee et al., 2023)							
WKIngram-25	12,659	47	41,873	3,228	74	3,391	1,131
WKIngram-50	12,022	72	82,481	9,328	93	9,672	3,225
WKIngram-75	6,853	52	28,741	2,722	65	3,430	1,144
WKIngram-100	9,784	67	49,875	12,136	37	13,487	4,496
InGram – NL (Lee et al., 2023)							
NLIngram-0	1,814	134	7,796	2,026	112	2,287	763
NLIngram-25	4,396	106	17,578	2,146	120	2,230	744
NLIngram-50	4,396	106	17,578	2,335	119	2,576	859
NLIngram-75	2,607	96	11,058	1,578	116	1,818	607
NLIngram-100	1,258	55	7,832	1,709	53	2,378	793
MTDEA – WikiTopics (Zhou et al., 2023)							
WikiTopicsMT1-tax	10,000	10	17,178	10,000	9	16,526	1,834
WikiTopicsMT1-health	10,000	7	14,371	10,000	7	14,110	1,566
WikiTopicsMT2-sci	10,000	16	16,471	10,000	16	14,852	1,650
WikiTopicsMT2-org	10,000	10	23,233	10,000	11	21,976	2,441
WikiTopicsMT3-art	10,000	45	27,262	10,000	45	28,023	3,113
WikiTopicsMT3-infra	10,000	24	21,990	10,000	27	21,646	2,405
WikiTopicsMT4-sci	10,000	42	12,576	10,000	42	12,516	1,388
WikiTopicsMT4-health	10,000	21	15,539	10,000	20	15,337	1,703
MTDEA – Other (Zhou et al., 2023)							
Metafam	1,316	28	13,821	656	28	7,257	184
FBNELL	4,636	100	10,275	4,752	183	10,685	597

Inductive-E target datasets. The *Inductive-E* setting contains 16 target graphs in which inference involves unseen entities while the relation vocabulary remains fixed. We use 10 GraIL variants (Teru et al., 2020), four HM/INDIGO graphs (Hamaguchi et al., 2017; Liu et al., 2021), and two ILPC2022 graphs (Galkin et al., 2022). FB15k-237 Inductive v1 and NELL Inductive v1 are excluded from the target set because they are used for pretraining. The dataset statistics are reported in Table D.4.

Table D.4: Target KGs and their statistics in the *Inductive-E* setting.

Dataset	Rel.	Training Graph		Validation Graph			Test Graph		
		Ent.	Triples	Ent.	Triples	Valid	Ent.	Triples	Test
GraIL – FB15k-237 (Teru et al., 2020)									
FB15k-237 Inductive v2	200	2,608	9,739	2,608	9,739	1,166	1,660	4,145	947
FB15k-237 Inductive v3	215	3,668	17,986	3,668	17,986	2,194	2,501	7,406	1,731
FB15k-237 Inductive v4	219	4,707	27,203	4,707	27,203	3,352	3,051	11,714	2,840
GraIL – WN18RR (Teru et al., 2020)									
WN18RR Inductive v1	9	2,746	5,410	2,746	5,410	630	922	1,618	373
WN18RR Inductive v2	10	6,954	15,262	6,954	15,262	1,838	2,757	4,011	852
WN18RR Inductive v3	11	12,078	25,901	12,078	25,901	3,097	5,084	6,327	1,143
WN18RR Inductive v4	9	3,861	7,940	3,861	7,940	934	7,084	12,334	2,823
GraIL – NELL (Teru et al., 2020)									
NELL Inductive v2	88	2,564	8,219	2,564	8,219	922	2,086	4,586	935
NELL Inductive v3	142	4,647	16,393	4,647	16,393	1,851	3,566	8,048	1,620
NELL Inductive v4	76	2,092	7,546	2,092	7,546	876	2,795	7,073	1,447
HM/INDIGO (Hamaguchi et al., 2017; Liu et al., 2021)									
HM-INDIGO	229	12,721	121,601	12,797	121,601	14,121	14,775	250,195	14,904
HM-5k	11	28,601	57,601	28,744	57,601	900	23,792	48,425	2,124
HM-3k	11	32,118	71,097	32,250	71,097	1,201	19,218	38,285	1,349
HM-1k	11	36,237	93,364	36,311	93,364	1,771	9,899	18,638	476
ILPC2022 (Galkin et al., 2022)									
ILPC2022-Large	65	46,626	202,446	29,246	77,044	10,179	29,246	77,044	10,184
ILPC2022-Small	48	10,230	78,616	6,653	20,960	2,908	6,653	20,960	2,902

Transductive target datasets. The *Transductive* setting contains 15 target graphs in which the entity and relation vocabularies are shared between target-graph training and inference. Specifically, we include FB15k-237 (Toutanova & Chen, 2015), WN18RR (Dettmers et al., 2018), CoDEx-M and CoDEx-L (Safavi & Koutra, 2020), WDsinger, NELL23k, three subsampled FB15k-237 variants (Lv et al., 2020), NELL-995 (Xiong et al., 2017), ConceptNet100k (Malaviya et al., 2020), AristoV4 (Chen et al., 2021), YAGO3-10 (Mahdisoltani et al., 2014), DBpedia100k (Ding et al., 2018), and Hetionet (Himmelstein et al., 2017). CoDEx-S is excluded from the target set because it is used for pretraining. The dataset statistics are reported in Table D.5.

Table D.5: Target KGs and their statistics in the *Transductive* setting.

Dataset	Entities	Relations	Train	Valid	Test
Standard KGC benchmarks					
FB15k-237 (Toutanova & Chen, 2015)	14,541	237	272,115	17,535	20,466
WN18RR (Dettmers et al., 2018)	40,943	11	86,835	3,034	3,134
NELL-995 (Xiong et al., 2017)	74,536	200	149,678	543	2,818
YAGO3-10 (Mahdisoltani et al., 2014)	123,182	37	1,079,040	5,000	5,000
CoDEx benchmarks (Safavi & Koutra, 2020)					
CoDEx-M	17,050	51	185,584	10,310	10,311
CoDEx-L	77,951	69	551,193	30,622	30,622
Sparse-KG benchmarks (Lv et al., 2020)					
WDsinger	10,282	135	16,142	2,163	2,203
NELL23k	22,925	200	25,445	4,961	4,952
FB15k-237-10	11,512	237	27,211	15,624	18,150
FB15k-237-20	13,166	237	54,423	16,963	19,776
FB15k-237-50	14,149	237	136,057	17,449	20,324
Additional diverse-domain benchmarks					
ConceptNet100k (Malaviya et al., 2020)	78,334	34	100,000	1,200	1,200
AristoV4 (Chen et al., 2021)	44,949	1,605	242,567	20,000	20,000
DBpedia100k (Ding et al., 2018)	99,604	470	597,572	50,000	50,000
Hetionet (Himmelstein et al., 2017)	45,158	24	2,025,177	112,510	112,510

D.1.2 BASELINES

Baseline families. We consider two major families of knowledge graph foundation models (KGFMs): *structure-based KGFMs* and *LLM-enhanced KGFMs*. Structure-based KGFMs achieve cross-graph generalization primarily by learning transferable structural patterns shared across different KGs, whereas LLM-enhanced KGFMs additionally exploit semantic knowledge or reasoning capabilities from pretrained language models.

Implementation. To ensure a controlled comparison, we retrain all baselines on the same three source KGs, namely FB15k-237 Inductive v1, NELL Inductive v1, and CoDEx-S, and evaluate them on identical target datasets and data splits. Unless otherwise specified, we follow the official implementations and recommended hyperparameter settings of each method, while adapting their data pipelines when necessary to comply with our unified preprocessing and evaluation protocols.

Structure-based KGFMs.

- **ULTRA** (Galkin et al., 2024) learns transferable relation representations through relation-level structural modeling and performs query-conditioned reasoning over the target KG.
- **KG-ICL** (Cui et al., 2024) formulates KG reasoning from an in-context learning perspective, using demonstrations from source KGs to support reasoning on unseen target graphs.
- **TRIX** (Zhang et al., 2025) performs transferable KG reasoning through query-conditioned structural modeling, emphasizing relational patterns that can be reused across different graphs.
- **MOTIF** (Huang et al., 2025a) captures recurring relational motifs as transferable structural patterns for cross-KG reasoning.
- **FLOCK** (Kim et al., 2026) models transferable graph structures through walk-based representations that encode local relational patterns across heterogeneous KGs.

LLM-enhanced KGFMs.

- **SEMMA** (Arun et al., 2025) augments structural KG reasoning with LLM-derived relation semantics, constructing a textual relation graph from enriched relation descriptions and fusing it with a structural relation graph for zero-shot link prediction on unseen KGs.

- **ProLink** (Wang et al., 2024) leverages a frozen LLM to translate relation semantics into structural prompt graphs that augment a pretrained GNN reasoner for low-resource inductive reasoning across unseen KGs.
- **KRLM** (Zhuo et al., 2026) coordinates intrinsic LLM knowledge with structural KG context through a dedicated knowledge-reasoning language and structure-aware prediction mechanism for inductive reasoning on unseen KGs.

D.1.3 Evaluation Protocol

We use a unified evaluation protocol across all methods to ensure that performance differences are not caused by differences in dataset construction or evaluation procedures. All methods use the same dataset inclusion policy, target splits, and processed graph instances. Whenever official training, validation, and test partitions are provided by a benchmark, we preserve these partitions.

For inductive datasets, the training graph and the inference graphs are kept separate according to the original benchmark definitions. In the *Inductive-e* setting, inference graphs contain entities that are unseen during training while using the same relation vocabulary. In the *Inductive-E,R* setting, both previously unseen entities and previously unseen relation types may occur in the inference graphs. For transductive datasets, training, validation, and test triples share the same entity and relation vocabularies.

All baselines are evaluated on exactly the same processed target graphs as our method. When an official implementation requires a method-specific input format, we only convert the common processed graph into the required representation and do not modify the underlying train/validation/test partition. This ensures that every method receives the same observable graph information and is evaluated on the same prediction queries.

D.1.4 Implementation Details

We use a hidden dimension of 48 and a six-layer query-conditioned GNN. For each query relation, we sample 7 support facts and extract 32 four-hop support chains from each support context. For entity-side reasoning, we sample up to 64 four-hop query-rooted chains. The word-level and chain-level codebooks contain 64 and 16 entries, respectively. The context, chain, and query-chain Transformers contain 1, 4, and 2 layers, respectively. For training, we use Adam with a learning rate of 1.5×10^{-4}, weight decay 10^{-4}, batch size 128, and $\lambda_{\mathrm{vq}} = 0.04$. Models are trained for at most 100 epochs with early stopping patience 5. All experiments are conducted on NVIDIA H200 GPUs.

D.2 Detailed Results for RQ1

To provide a more comprehensive view of RQ1, we report the full entity prediction results of ChainGram and representative KG foundation models across all evaluated KGs. Due to space limitations, we select KRLM and FLOCK as representative baselines from the LLM-enhanced and structure-oriented KGFM paradigms, respectively. We consider three reasoning settings: Inductive-E, Inductive-E,R, and Transductive, and report MRR and Hits@10 under both zero-shot and fine-tuned settings.

Table D.6 presents the complete entity prediction results under both zero-shot and fine-tuned settings. ChainGram achieves the best MRR on 40 of 54 datasets in the zero-shot setting, with an average improvement of about 10.8% over the strongest baseline. The gains are especially notable on datasets such as MT1-tax and DBpedia100k, and ChainGram remains consistently competitive after fine-tuning, where it further achieves an average improvement of about 7.8%. These results demonstrate the effectiveness and robustness of ChainGram across heterogeneous KGs. For AristoV4, the fine-tuned result of KRLM is unavailable due to an out-of-memory (OOM) error.

Figure D.1 provides an overall comparison of MRR under the zero-shot and fine-tuned settings. Together, these detailed results complement the RQ1 analysis in the main text and provide a complete comparison of ChainGram with SOTA KG foundation models across different KGs and reasoning settings.

Table D.6: Zero-shot and finetuned entity prediction results. Bold indicates the best score per row within each evaluation setting. * denotes results unavailable due to out-of-memory (OOM).

Dataset	Zero-shot						Finetuned					
	KRLM		FLOCK		CHAINGRAM		KRLM		FLOCK		CHAINGRAM	
	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
Inductive e, r												
WK-25	0.412	0.627	0.306	0.493	**0.429**	**0.641**	0.411	0.614	0.333	0.498	**0.431**	**0.641**
WK-50	0.260	0.416	**0.288**	**0.486**	0.271	0.441	0.261	0.434	0.144	0.277	**0.275**	**0.439**
WK-75	0.430	**0.612**	0.388	0.548	**0.452**	0.595	0.437	**0.613**	0.401	0.557	**0.457**	0.610
WK-100	0.225	0.379	0.176	0.289	**0.261**	**0.409**	0.239	0.408	0.186	0.303	**0.267**	**0.419**
FB-25	0.400	0.667	0.328	0.564	**0.418**	**0.680**	0.404	0.685	0.292	0.523	**0.430**	**0.687**
FB-50	0.352	0.571	0.288	0.486	**0.354**	**0.577**	0.356	**0.600**	0.356	0.570	**0.373**	0.582
FB-75	0.428	**0.641**	0.376	0.576	**0.435**	0.636	0.428	**0.660**	0.416	0.626	**0.438**	0.644
FB-100	0.479	0.683	0.449	0.651	**0.494**	**0.696**	0.471	0.697	0.453	0.661	**0.496**	**0.702**
NL-0	0.514	0.771	0.431	0.647	**0.570**	**0.790**	0.504	0.781	0.421	0.631	**0.567**	**0.790**
NL-25	**0.505**	**0.706**	0.437	0.637	0.504	0.694	0.503	0.720	0.415	0.611	**0.540**	**0.732**
NL-50	0.516	0.706	0.426	0.615	**0.530**	**0.710**	0.510	**0.713**	0.420	0.598	**0.527**	0.691
NL-75	0.445	0.681	0.392	0.595	**0.464**	**0.687**	0.422	0.671	0.378	0.572	**0.461**	**0.693**
NL-100	0.538	0.767	0.501	0.711	**0.553**	**0.776**	0.518	0.758	0.494	0.704	**0.557**	**0.787**
MT1 tax	0.225	0.334	0.259	0.329	**0.364**	**0.485**	0.398	0.516	0.278	0.346	**0.418**	**0.532**
MT1 health	0.280	0.441	0.364	0.456	**0.380**	**0.487**	0.360	0.470	0.386	**0.487**	**0.396**	0.481
MT2 sci	0.256	0.363	0.280	0.384	**0.328**	**0.448**	0.328	**0.531**	0.364	0.524	**0.385**	0.517
MT2 org	0.086	0.151	**0.092**	**0.161**	**0.092**	0.154	0.084	0.168	0.091	0.158	**0.102**	**0.171**
MT3 art	0.216	0.374	0.260	**0.410**	**0.267**	0.403	0.297	**0.474**	0.311	0.471	**0.314**	0.466
MT3 infra	0.599	0.750	**0.648**	**0.797**	**0.648**	0.778	0.645	0.820	**0.704**	**0.836**	0.697	0.826
MT4 sci	0.264	0.447	**0.303**	**0.464**	**0.303**	**0.464**	0.288	0.477	**0.321**	**0.507**	0.312	0.477
MT4 health	0.658	**0.776**	0.645	0.755	**0.671**	0.771	0.676	**0.790**	0.667	0.783	**0.712**	0.783
Metafam	0.174	0.535	**0.486**	0.913	0.458	**0.940**	**1.000**	**1.000**	0.999	**1.000**	**1.000**	**1.000**
FBNELL	0.497	0.688	0.501	0.683	**0.520**	**0.698**	0.500	0.672	0.502	**0.706**	**0.524**	0.696
Inductive e												
HM Indigo	0.383	0.559	**0.391**	**0.589**	0.352	0.553	**0.414**	**0.612**	0.401	0.591	0.387	0.593
HM 5k	0.033	0.075	0.048	**0.117**	**0.056**	0.090	0.044	0.079	**0.072**	**0.135**	0.059	0.093
HM 3k	0.037	0.085	0.052	**0.126**	**0.059**	0.098	0.047	0.077	**0.060**	**0.115**	0.057	0.091
HM 1k	0.060	0.124	0.052	**0.136**	**0.065**	0.095	**0.059**	**0.130**	0.044	0.100	0.052	0.080
ILPC2022 Large	0.234	0.406	**0.294**	0.409	0.278	**0.417**	0.294	0.427	0.314	0.424	**0.327**	**0.446**
ILPC2022 Small	0.295	**0.458**	0.275	0.406	**0.300**	**0.458**	0.320	**0.470**	0.299	0.435	**0.321**	0.469
FB-v2	**0.546**	**0.749**	0.538	0.720	0.539	0.721	0.552	0.729	**0.566**	**0.731**	0.549	0.723
FB-v3	**0.512**	**0.683**	0.500	0.655	0.509	0.668	0.521	0.677	**0.530**	0.668	0.515	**0.671**
FB-v4	**0.496**	**0.686**	0.467	0.641	0.490	0.674	**0.507**	**0.686**	0.498	0.656	0.504	0.682
WN-v1	**0.677**	**0.776**	0.645	0.765	0.670	0.773	0.706	0.794	0.669	0.775	**0.717**	**0.806**
WN-v2	**0.667**	0.761	0.661	0.765	0.661	**0.773**	0.685	0.776	0.685	0.774	**0.702**	**0.790**
WN-v3	0.387	0.507	**0.404**	**0.539**	0.399	0.526	0.452	0.571	**0.455**	**0.583**	0.445	0.555
WN-v4	**0.624**	0.695	0.596	**0.702**	0.603	0.700	**0.661**	0.730	0.638	0.704	0.659	**0.738**
NL-v2	0.572	0.783	0.581	0.778	**0.599**	**0.794**	0.578	0.782	**0.607**	**0.801**	0.606	0.794
NL-v3	0.571	0.769	0.579	0.765	**0.615**	**0.786**	0.602	0.783	0.598	0.775	**0.610**	**0.785**
NL-v4	0.499	0.768	0.553	0.760	**0.569**	**0.772**	0.537	0.761	**0.577**	**0.780**	0.567	0.771
Transductive												
FB15k-237	0.278	0.439	0.178	0.252	**0.347**	**0.514**	0.305	0.506	0.338	0.523	**0.394**	**0.559**
WN18RR	0.424	0.496	0.402	0.491	**0.459**	**0.535**	0.512	0.629	0.537	0.647	**0.570**	**0.673**
CoDEx Medium	0.312	0.466	0.200	0.306	**0.327**	**0.482**	0.333	0.488	0.346	0.487	**0.384**	**0.534**
WDsinger	0.381	0.507	0.387	0.496	**0.478**	**0.593**	0.422	0.530	0.413	0.528	**0.495**	**0.613**
NELL23k	0.243	0.426	0.239	0.403	**0.276**	**0.457**	0.262	0.444	0.274	0.457	**0.285**	**0.469**
FB15k-237(10)	0.159	0.273	0.145	0.249	**0.172**	**0.292**	0.168	0.290	0.170	0.293	**0.175**	**0.297**
FB15k-237(20)	0.169	0.292	0.171	0.296	**0.186**	**0.319**	0.182	0.314	0.191	0.327	**0.198**	**0.338**
FB15k-237(50)	0.205	0.353	0.191	0.332	**0.233**	**0.389**	0.219	0.386	0.224	0.388	**0.249**	**0.413**
NELL995	0.419	0.580	**0.518**	**0.659**	0.514	0.627	0.500	0.639	0.523	0.666	**0.547**	**0.668**
ConceptNet100k	0.066	0.149	0.209	0.388	**0.288**	**0.503**	0.302	0.524	0.340	0.564	**0.390**	**0.608**
AristoV4	0.228	**0.343**	0.225	0.326	**0.232**	0.342	–*	–*	0.339	0.489	**0.452**	**0.571**
YAGO310	0.388	0.586	0.335	0.515	**0.401**	**0.601**	0.534	0.701	0.493	0.655	**0.583**	**0.720**
CoDEx Large	0.249	0.391	**0.302**	0.426	0.297	**0.432**	0.322	0.455	0.318	0.434	**0.358**	**0.483**
DBpedia100k	0.276	0.460	0.383	0.569	**0.432**	**0.595**	0.399	0.587	0.384	0.541	**0.512**	**0.651**
Hetionet	0.155	0.306	0.209	0.331	**0.277**	**0.406**	0.247	0.450	0.383	0.523	**0.427**	**0.551**

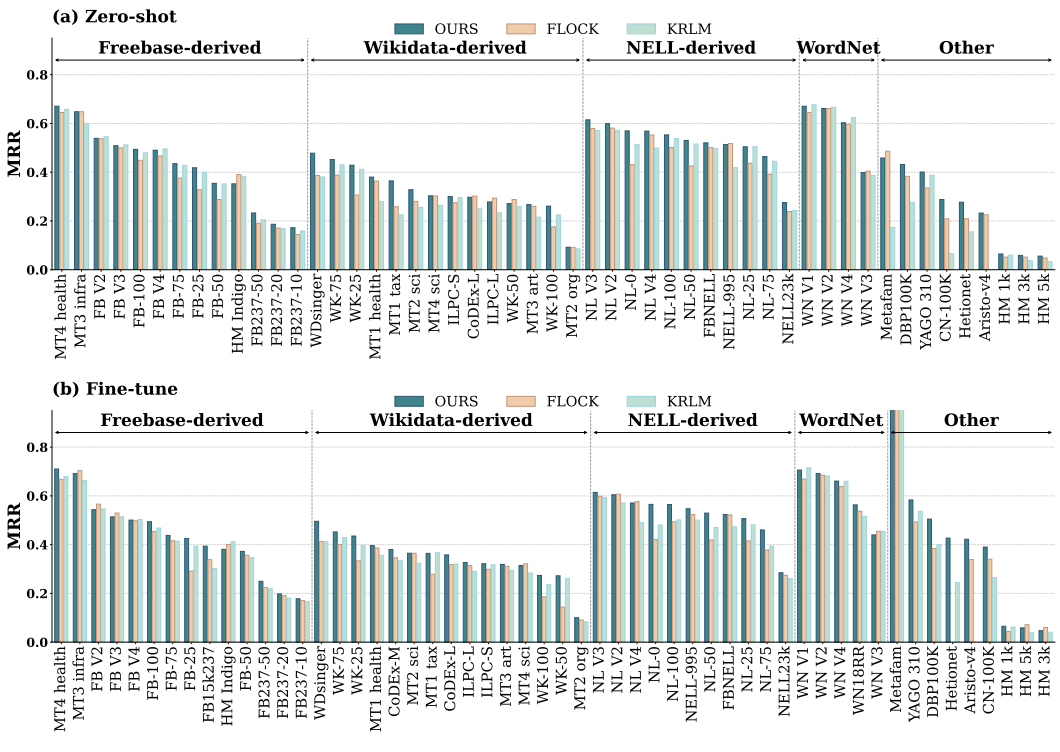


Figure D.1: Zero-shot and fine-tune MRR (higher is better) in the entity prediction task

D.3 Detailed Analysis of Learned Structural Grammar (RQ2)

D.3.1 Experimental Settings

For the real-KG analyses in RQ2, we use a fixed mechanism-probe set containing 24 KGs across three transfer settings. The datasets are organized as follows:

Inductive-E (8 KGs):
HM:indigo, FB15k237Inductive:v2, FB15k237Inductive:v3, WN18RRInductive:v1, WN18RRInductive:v3, NELLInductive:v2, NELLInductive:v4, and ILPC2022:small.

Inductive-E,R (8 KGs):
WKIngram:100, WKIngram:75, FBIngram:100, NLIngram:75, NLIngram:100, WikiTopicsMT4:sci, Metafam, and FBNELL.

Transductive (8 KGs):
WDsinger, NELL23k, FB15k237_10, FB15k237_20, FB15k237_50, CoDExLarge, FB15k237, and CoDExMedium.

Unless otherwise specified, all real-KG analyses use frozen checkpoints trained with three random seeds. The word-level and chain-level codebooks contain 64 and 16 entries, respectively.

D.3.2 Prototype Organization Analysis

We examine whether training organizes continuous chain representations around the learned structural prototypes. The analysis compares a randomly initialized encoder with the trained ChainGram encoder under the same architecture and input chains.

Protocol. For each of the 24 KGs, we sample 512 anonymous structural chains and encode the same chains using two encoders: (1) a randomly initialized encoder constructed with the same configuration as ChainGram, and (2) the trained encoder from seed 1024. The representations

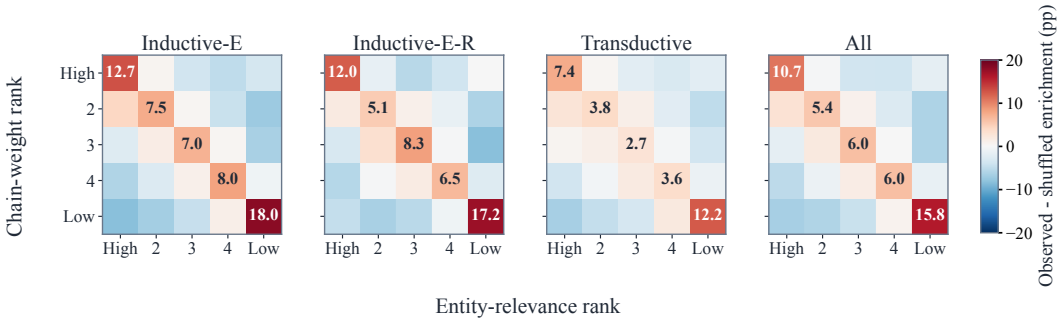


Figure D.2: Detailed chain-weight / entity-relevance alignment across transfer settings. Positive diagonal enrichment indicates agreement between learned chain importance and entity relevance.

are colored according to the chain-level code assignments generated by the trained encoder. The visualization therefore examines whether training reshapes the continuous representation space to align with the learned discrete prototypes. It does not evaluate agreement with external semantic labels, since the prototype assignments are induced by the model itself.

Visualization. For visualization, we randomly sample 80 chains from each KG, resulting in 1,920 representations. We apply t-SNE separately to the randomly initialized and trained representations with perplexity 40, 1,500 iterations, and a fixed random seed.

Organization Metric. Following the silhouette coefficient (Rousseeuw, 1987), we quantify prototype organization in the original representation space using a balanced cosine silhouette score. Given a chain representation $\mathbf{h}_i$ with prototype assignment y_i, we define a_i as the average cosine distance to samples assigned to the same prototype and b_i as the minimum average cosine distance to samples assigned to other prototypes. The silhouette score is computed as

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}. \tag{38}$$

We report the mean score with equal weighting over prototypes to avoid domination by frequently assigned codes. As shown in Fig. 4(a), the balanced cosine silhouette increases from 0.0837 for the randomly initialized encoder to 0.3924 after training, yielding an absolute gain of 0.3087. This indicates that training substantially improves the alignment between continuous chain representations and the learned structural prototypes.

D.3.3 Chain-Entity Relevance Analysis

Motivation. The Grammar-Guided Entity Reasoner assigns different weights to query-rooted chains when aggregating chain evidence. We analyze whether these learned weights correspond to the actual contribution of each chain to entity prediction, i.e., whether highly weighted chains provide stronger evidence for identifying the correct entity.

Chain Contribution Measurement. For a query q, let $\mathbf{h}_i$ denote the representation of the i-th query-rooted chain and $\mathbf{e}_v$ denote the candidate entity representation. Following the original entity-side matcher in the frozen ChainGram model, the contribution of an individual chain is computed as

$$s_i(v) = \frac{(\mathbf{W}_c \mathbf{h}_i)^\top (\mathbf{W}_e \mathbf{e}_v)}{\sqrt{d}}. \tag{39}$$

This score directly corresponds to the chain-wise linear contribution in the original prediction function and does not introduce additional training or probing models.

For the ground-truth entity v^+, we define the relevance of chain i as

$$R_i = s_i(v^+) - \frac{1}{|\mathcal{N}_q|} \sum_{v^- \in \mathcal{N}_q} s_i(v^-), \tag{40}$$

where $\mathcal{N}_q$ denotes the sampled filtered negative entities. A larger R_i indicates that the chain provides stronger evidence for separating the correct entity from negative candidates.

Weight–Relevance Alignment. For each query, we rank chains according to their learned weights ω_i and relevance scores R_i, and divide both rankings into five approximately equal-frequency bins. To remove the effect of the marginal weight distribution, we compare the observed joint ranking distribution with a within-query shuffled control, where chain weights are randomly permuted while preserving chain representations, relevance scores, and candidate entities.

The enrichment score is computed as

$$\Delta(j,k) = 100\,(P_{\text{obs}}(R=k|W=j) - P_{\text{shuf}}(R=k|W=j))\,, \tag{41}$$

where W and R denote the chain-weight and relevance ranks, respectively.

Results. The main-paper heatmap summarizes the overall alignment across 24 KGs for the selected checkpoints. All diagonal cells exhibit positive enrichment over the shuffled baseline, with gains ranging from 5.4 to 15.8 percentage points, indicating that chains receiving higher weights from the reasoner tend to provide stronger evidence for identifying the correct entity.

Figure D.2 further reports the results under different transfer settings. The diagonal enrichment consistently appears in Inductive-E, Inductive-E,R, and Transductive settings, showing that the learned chain importance is aligned with entity relevance across diverse reasoning scenarios.

D.3.4 Codebook Utilization Analysis

Codebook Utilization. We first examine whether the learned codebooks are effectively utilized. As shown in Fig. D.3(a), the word-level codebook activates nearly all 64 entries, while the chain-level codebook typically activates 11–12 out of 16 entries across different transfer settings. Both levels also maintain high normalized assignment entropy, indicating that the learned structural vocabulary does not collapse to a few dominant codes and captures diverse structural patterns.

Codebook Utility. We further test whether the learned structure–code correspondence contributes to prediction by permuting quantized representations across structural instances while preserving code frequencies and model parameters. As shown in Fig. D.3(b), disrupting the word-level and chain-level alignment causes a clear MRR drop, and jointly perturbing both levels produces a similar degradation. Together with the chain-level ablation in Sec. 4.4, these results show that the hierarchical codebooks provide useful structural abstractions for reasoning.

D.3.5 Cross-KG Pattern Transferability Analysis

We further examine whether ChainGram preserves KG-invariant structural patterns across unseen graphs.

Controlled Synthetic Probe. Real KGs vary simultaneously in relation identities, frequencies, topology, and entity distributions, while ground-truth correspondence between structural patterns is unavailable. We therefore construct a controlled synthetic benchmark with three four-hop patterns:

$$\mathcal{P} = \{(A,A,A,B), (A,A,B,A), (A,B,A,A)\}, \tag{42}$$

denoted as AAAB, AABA, and ABAA. All three patterns have the same topology, direction, chain length, and relation multiset (three As and one B), differing only in the position of B. Each KG maps A and B to its own relation identities and uses a disjoint entity/relation namespace, preventing identity or frequency shortcuts. The benchmark contains 12 training, 6 validation, and 12 unseen test KGs, with 64 episodes per pattern and eight support examples per episode.

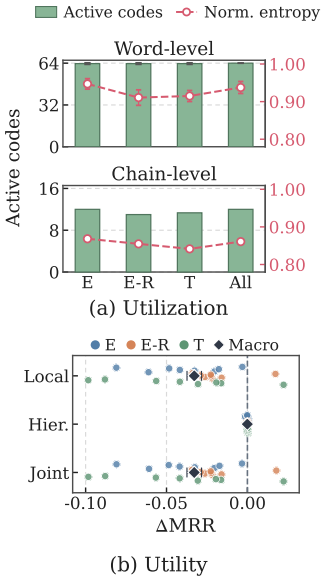


Figure D.3: Analysis of code utilization and predictive effect.

Pattern Consistency Score. For an anchor representation $\mathbf{z}^a$ from one unseen KG, we select a positive $\mathbf{z}^+$ from another KG with

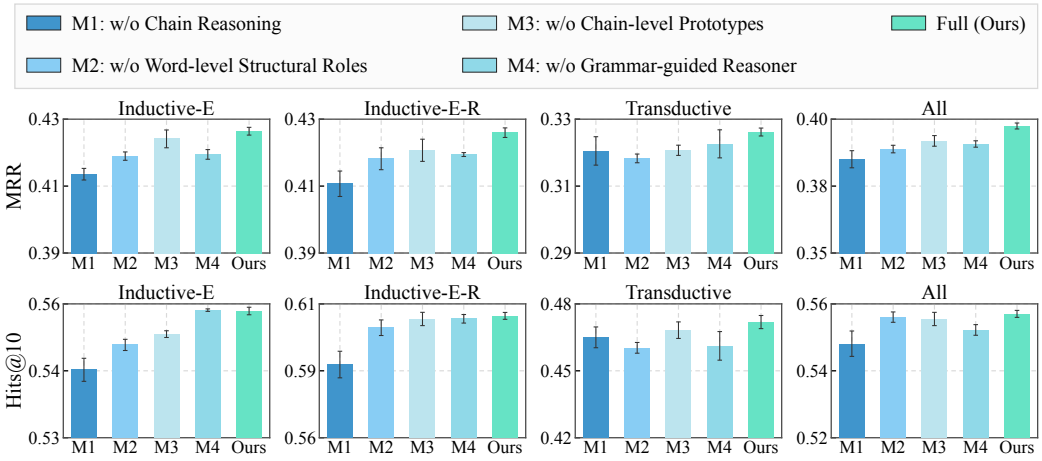


Figure D.4: Detailed ablation results under different transfer settings. We report both MRR and Hits@10 for the four main variants.

the same pattern and a hard negative $\mathbf{z}^-$ from the same target KG but with a different pattern. We define

$$m_i = \cos(\mathbf{z}_i^a, \mathbf{z}_i^+) - \cos(\mathbf{z}_i^a, \mathbf{z}_i^-), \qquad \text{PCS} = \frac{1}{N}\sum_{i=1}^{N} m_i. \tag{43}$$

Because the positive and hard negative share the same target KG, topology, and relation-frequency profile, a higher PCS indicates that the representation better preserves the underlying structural pattern across KGs rather than graph-specific identities or frequency statistics.

Results. The upper panel of Fig. 4(c) projects each representation according to its similarity to the three pattern prototypes; movement toward a labeled direction therefore indicates stronger affinity to that pattern. CHAINGRAM forms three clearly separated directional groups, whereas the representations of ULTRA, KG-ICL, and MOTIF are much more concentrated and entangled. This observation is consistent with our motivation that implicit graph representations can mix different structural patterns. Quantitatively, CHAINGRAM achieves a PCS of 0.0476, substantially higher than ULTRA, KG-ICL, and MOTIF, confirming stronger cross-KG preservation of structural distinctions.

D.4 DETAILED ANALYSIS OF ABLATION STUDY (RQ3)

We conduct ablation experiments to examine which components contribute to the transferable reasoning ability of CHAINGRAM. We consider the following variants:

- **M1: w/o Chain Reasoning**, which removes the explicit chain-based reasoning branch and relies only on graph-level representations for prediction.
- **M2: w/o Word-Level Structural Roles**,which removes the word-level codebook and replaces discrete local structural roles with continuous representations.
- **M3: w/o Chain-Level Prototypes**, which removes the chain-level codebook and represents chain-level compositions using continuous representations.
- **M4: w/o Grammar Guided Reasoner**, which removes the grammar-guided chain matching module and relies on the base GNN prediction.

The main paper reports the core ablation results using MRR under the zero-shot protocol. Here, we provide a more complete analysis with more metric Hits@10 and additional ablation settings to further examine the contribution of each component.

Main Component Ablations. Figure D.4 complements the main-paper analysis with Hits@10 results. The overall trends are consistent with those observed for MRR. Removing explicit chain reasoning (**M1**) produces the most pronounced degradation across settings, confirming that query-rooted chain evidence provides information beyond graph-level message passing. Removing either

Table D.7: Zero-shot entity prediction results under the alternative three-KG pretraining configuration. Best and second-best results are shown in **bold** and underline, respectively.

Model	Inductive-E,R (23)		Inductive-E (18)		Transductive (13)		Overall (54)	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
ULTRA	0.3434	0.5112	0.4157	0.5676	0.2918	0.4263	0.3551	0.5095
KG-ICL	0.4133	0.5953	0.4486	0.5818	0.3502	0.4955	0.4099	0.5668
TRIX	0.3378	0.5153	0.4042	0.5453	0.2752	0.4127	0.3449	0.5006
MOTIF	0.3491	0.5254	0.4363	0.5770	0.2983	0.4347	0.3659	0.5207
FLOCK	0.3749	0.5500	0.4146	0.5544	0.2728	0.4026	0.3635	0.5160
CHAINGRAM	**0.4206**	**0.6050**	**0.4540**	**0.5869**	**0.3573**	**0.5048**	**0.4165**	**0.5748**

word-level structural roles (**M2**) or chain-level prototypes (**M3**) also reduces overall performance, supporting the hierarchical abstraction of local structural roles and chain-level compositions. Finally, removing the grammar-guided reasoner (**M4**) generally weakens performance, indicating that learned structural patterns are most effective when explicitly matched with query-specific chain evidence. Although individual variants can be competitive on isolated metrics, the complete CHAINGRAM provides the strongest and most consistent performance overall.

D.5 DETAILED ANALYSIS OF PRETRAINING AND KNOWLEDGE SOURCES (RQ4)

This section provides the detailed experimental settings and complete results for the three analyses in Sec. 4.5, including pretraining KG distribution, pretraining scale, and comparison with LLM-enhanced KGFMs.

D.5.1 ROBUSTNESS TO PRETRAINING KG DISTRIBUTION

Experimental Setup. To examine whether the gains of CHAINGRAM depend on the specific pretraining KGs used in the main experiments, we adopt the commonly used three-KG pretraining configuration consisting of FB15k-237, WN18RR, and CoDEx-Medium, following prior KGFM studies. For baselines with results already reported under this configuration, we use the corresponding published results from the MOTIF and FLOCK evaluations; KG-ICL and CHAINGRAM are retrained under the same source configuration and evaluation protocol. All methods are evaluated zero-shot on the same 54 target KGs.

Results. Table D.7 shows that CHAINGRAM achieves the best performance on all eight aggregate metrics under this alternative pretraining distribution. Compared with the strongest baseline, CHAINGRAM improves overall MRR and Hits@10 by 1.61% and 1.41%, respectively. The gains are also consistent across transfer settings, with MRR improvements of 1.77% on Inductive-E,R, 1.20% on Inductive-E, and 2.03% on Transductive. These results indicate that the advantage of CHAINGRAM is not tied to the particular source KGs used in the main experiments, but remains under a different and widely used KGFM pretraining configuration.

D.5.2 EFFECT OF PRETRAINING SCALE

Experimental Setup. To examine how the scale and diversity of the pretraining corpus affect cross-KG transfer, we progressively expand a nested collection of source KGs from one to seven graphs while keeping the model architecture, training protocol, and evaluation targets fixed. Specifically, we consider five configurations:

$$\begin{aligned} &K1 : \{\text{FB237-v1}\}, \\ &K2 : K1 + \{\text{NELL-v1}\}, \\ &K3 : K2 + \{\text{CoDEx-S}\}, \\ &K5 : K3 + \{\text{WN18RR}, \text{WikiTopicsMT4-sci}\}, \\ &K7 : K5 + \{\text{ConceptNet100k}, \text{WDsinger}\}. \end{aligned} \tag{44}$$

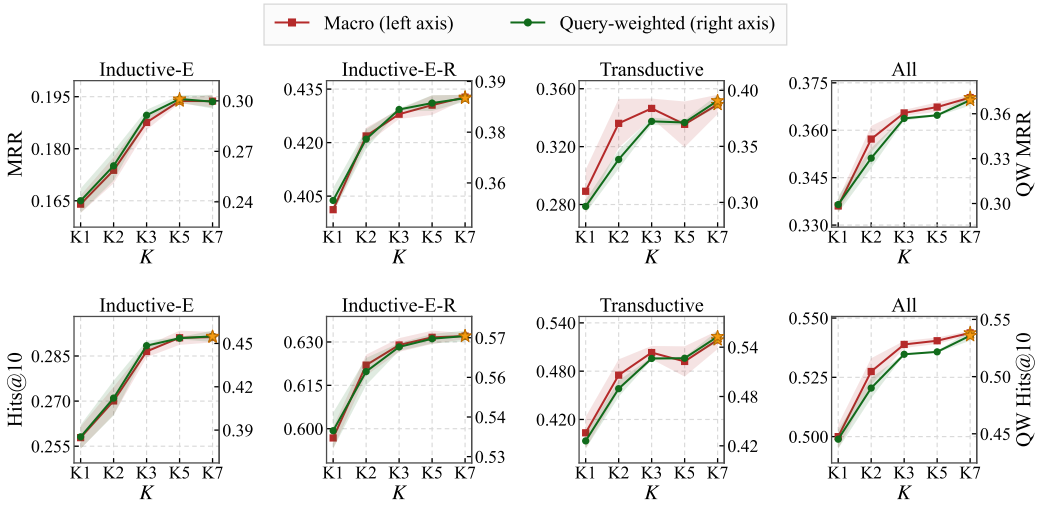


Figure D.5: Effect of pretraining scale on transfer performance. $K1$–$K7$ denote progressively expanded source-KG collections. Red curves report KG-level macro averages (left axis), while green curves report query-weighted averages (right axis). The upper and lower rows show MRR and Hits@10, respectively.

To ensure a fair comparison across different pretraining scales, all configurations are evaluated on the same frozen set of 27 target KGs. The set contains 6 Inductive-E, 18 Inductive-E,R, and 3 Transductive KGs:

- **Inductive-E:** HM:indigo, ILPC2022:large, HM:5k, HM:3k, ILPC2022:small, and HM:1k.
- **Inductive-E,R:** WKIngram:100/25/50/75, FBIngram:25/50/75/100, NLIngram:0/25/50/75/100, Metafam, FBNELL, WikiTopicsMT2:org, WikiTopicsMT3:art, and WikiTopicsMT3:infra.
- **Transductive:** AristoV4, YAGO310, and DBpedia100k.

Using a common target set is important because the source collection changes from $K1$ to $K7$. If each configuration were evaluated on its own set of unseen graphs, some KGs could be treated as test graphs for a smaller K but become pretraining graphs for a larger K, making the results incomparable. Our protocol therefore keeps the evaluation objects fixed and varies only the pretraining collection.

Evaluation and Aggregation. All models are evaluated on the test split using filtered entity ranking with MRR and Hits@10. We report both KG-level macro averages and query-weighted averages. For pretraining configuration K and seed s, the macro metric is

$$M_{K,s}^{\text{Macro}} = \frac{1}{27} \sum_{g=1}^{27} M_{K,s,g}, \tag{45}$$

where each target KG contributes equally. The query-weighted metric is

$$M_{K,s}^{\text{QW}} = \frac{\sum_g n_{s,g} M_{K,s,g}}{\sum_g n_{s,g}}, \tag{46}$$

where $n_{s,g}$ denotes the number of evaluated queries in KG g. The two aggregations are complementary because the target KGs differ substantially in query count. Results are averaged over the three training seeds, with population standard deviation used to summarize seed variability.

Results. As shown in Fig. D.5, increasing the pretraining collection generally improves transfer performance across both macro and query-weighted evaluations. From $K1$ to $K7$, overall macro MRR improves by 10.2%, while Hits@10 improves by 8.7%. The query-weighted results exhibit a similar overall trend, indicating that the gains are not caused solely by the equal weighting of target KGs.

The effect varies across transfer settings. Inductive-E improves substantially from $K1$ to $K5$ and then approaches saturation, whereas Inductive-E,R shows a steadier improvement as additional source KGs are introduced. The Transductive setting exhibits greater variation, including a temporary decrease at $K5$, but recovers at $K7$. Despite these setting-specific fluctuations, the overall macro performance increases consistently as the pretraining collection becomes broader.

D.5.3 Comparison with LLM-Enhanced KGFMs

Experimental Setup. LLM-enhanced KGFMs often exploit textual semantics associated with entities, but such information can be incomplete or unavailable in real-world KGs. To evaluate robustness to missing textual information, we divide the 54 evaluation benchmark configurations into two groups according to the availability of entity text. Specifically, the **Text-available** group contains 26 configurations with textual information, while the **Text-missing** group contains 28 configurations without such information. All evaluations in this analysis are conducted under the **zero-shot** setting, without finetuning on the target datasets.

In addition to this benchmark-level comparison, we conduct a controlled zero-shot ablation on seven Text-available datasets spanning the three reasoning settings:

- **Inductive-E,R:** FBIngram-100, WKIngram-100, and NLIngram-100.
- **Inductive-E:** FB15k-237 Inductive v4 and ILPC2022-Large.
- **Transductive:** WDsinger and CoDEx-M.

For each dataset, we remove the entity textual descriptions while keeping the KG structure, model configuration, and evaluation protocol unchanged. For each model, the same pretrained checkpoint is used in the original-text and text-missing conditions. The model is evaluated directly on each target KG, with no target-specific finetuning in either condition. This controlled comparison therefore isolates the effect of textual information from differences in graph structure, training adaptation, or evaluation setting.

Evaluation and Aggregation. We use MRR as the evaluation metric. For each model, we first compute the macro-average MRR separately over the Text-available and Text-missing groups. Let S_{avail} and S_{miss} denote the corresponding macro-average MRR values. The overall macro-average is computed from all valid benchmark results, or equivalently as the sample-count-weighted average of the two groups:

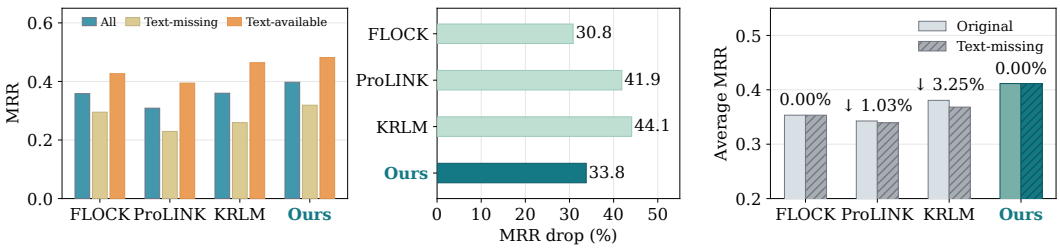
$$S_{\text{all}} = \frac{n_{\text{avail}} S_{\text{avail}} + n_{\text{miss}} S_{\text{miss}}}{n_{\text{avail}} + n_{\text{miss}}}. \tag{47}$$

For FLOCK, ProLINK, and ChainGram, the Text-available and Text-missing groups contain 26 and 28 valid results, respectively, giving 54 results in total. The KRLM result on AristoV4 is unavailable and is excluded from aggregation. Consequently, the KRLM Text-missing and overall averages are computed from 27 and 53 valid results, respectively.

We quantify the relative degradation associated with missing textual information as

$$\text{Drop} = \frac{S_{\text{avail}} - S_{\text{miss}}}{S_{\text{avail}}} \times 100\%. \tag{48}$$

A larger Drop indicates greater sensitivity to the absence of textual semantics. For the controlled ablation, we additionally compare the average MRR obtained with and without entity textual descriptions on the same seven datasets.



(a) MRR by text availability (b) Relative MRR drop without text (c) Text-removal robustness

Figure D.6: Robustness to missing textual information in zero-shot reasoning. (a) Macro-average MRR over all benchmark configurations and over the Text-available and Text-missing groups. (b) Relative MRR degradation from Text-available to Text-missing benchmarks. (c) Controlled entity-text removal on seven datasets.

Results. As shown in Fig. D.6(a) and (b), CHAINGRAM maintains strong performance across both Text-available and Text-missing KGs. Its relative MRR gap is 33.79%, close to the 30.83% gap of the structural baseline FLOCK. In comparison, ProLINK and KRLM exhibit larger gaps of 41.85% and 44.10%, respectively. Their average gap is approximately $1.27\times$ that of CHAINGRAM, indicating a stronger association between their performance and the availability of textual semantics.

The controlled experiment in Fig. D.6(c) provides a more direct evaluation of sensitivity to textual information. After removing entity textual descriptions, CHAINGRAM and FLOCK remain unaffected, as they do not rely on such textual inputs, whereas ProLINK and KRLM exhibit relative MRR drops of 1.03% and 3.25%, respectively. These results indicate that CHAINGRAM performs transferable reasoning solely from graph structure without requiring entity textual descriptions.