Efficient and Transferable Agentic Knowledge Graph RAG via Reinforcement Learning

Junhong Lin[1*] **Shicheng Liu**[1*] **Jinyeop Song**[1] **Song Wang**[2]
Julian Shun[1] **Yada Zhu**[3]

[1]MIT CSAIL [2]University of Central Florida [3]IBM Research

{junhong,liushch,yeopjin,jshun}@mit.edu song.wang@ucf.edu
yzhu@us.ibm.com

Abstract

Knowledge-graph retrieval-augmented generation (KG-RAG) couples large language models (LLMs) with structured, verifiable knowledge graphs (KGs) to reduce hallucination and provide reasoning traces. However, current KG-RAG systems often rely on fixed pipelines of multiple LLM modules (e.g., planning, reasoning, and responding), which inflate inference costs and tie performance to specific graph schemas. To address this, we introduce KG-R1, an agentic framework that optimizes KG-RAG through reinforcement learning. Unlike modular workflows, KG-R1 uses a single agent that interacts with KGs as its environment, learning to retrieve information at each step and incorporating it into its reasoning and generation in a unified process. Across Knowledge-Graph Question Answering benchmarks, KG-R1 demonstrates both *efficiency* and *transferability*—using Qwen 2.5-3B, KG-R1 improves answer accuracy with fewer generation tokens than prior multi-module workflow methods that use much larger foundation or fine-tuned models. Furthermore, KG-R1 exhibits strong *plug-and-play* capability: after training, it maintains accuracy on unseen KGs without retraining. These properties make KG-R1 a promising KG-RAG framework for real-world deployment. Our code is publicly available at github.com/junhongmit/KG-R1/.

1 Introduction

Retrieval-augmented generation (RAG) [1] has gained popularity as a way to enhance large language models (LLMs) with access to external knowledge, thereby reducing hallucination and improving accuracy in knowledge-intensive tasks. Recent research has extended this idea to knowledge graph retrieval-augmented generation (KG-RAG), where *knowledge graphs (KGs)* are leveraged as the retrieval source. A KG is a structured representation of knowledge in the form of entities (nodes) and their relationships (edges) that encodes factual knowledge in a graph format. Augmenting LLMs with KGs has proven effective not only in mitigating the knowledge bottleneck but also in improving reasoning performance over complex multi-hop relations and enhancing adaptation to continually evolving real-world information [2, 3, 4]. These properties make KG-RAG especially promising in high-stakes domains, such as medical consultation and legal analysis [5, 6].

As shown in Figure 1, a typical KG-RAG adopts a *modular workflow* consisting of four subtasks: *retrieval* to query facts from KGs, *reasoning* to process retrieved information, *reviewing* to verify logical consistency, and *responding* to synthesize the answer [7, 2, 3, 4]. Each subtask is handled by specialized LLM modules through two main methods: (i) prompt-based modules with task-specific instructions, often including in-context demonstrations [7]; and (ii) fine-tuned modules tailored to specific tasks (e.g., SPARQL generation [8, 9] or relation extraction [10]) on specific KGs.

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

*Equal contribution.

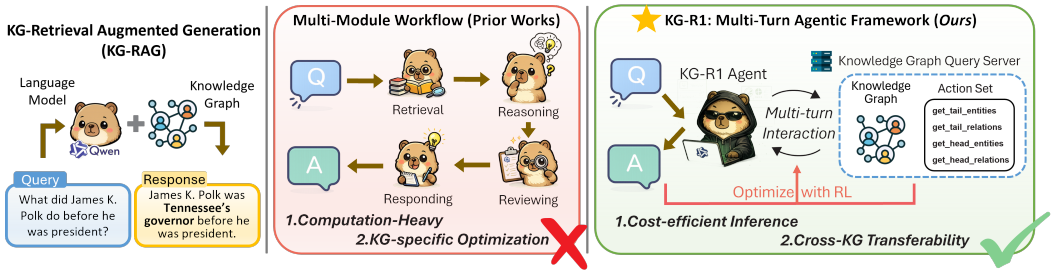


Figure 1: Overview of KG-R1, a multi-turn agentic KG-RAG framework. The framework enables cost-efficient inference and demonstrates strong cross-KG transferability.

Despite improved reasoning accuracy, the real-world deployment of such workflows faces two key challenges: achieving high reasoning **effectiveness** while maintaining **efficiency**, and **generalization to new or updated KGs**. First, prompt-based methods that repeatedly call large foundation LLMs accumulate inference across stages and drive up latency, token usage, and compute (e.g., ToG [2] and ReKnoS [4]; see left panel of Figure 2). Second, these prompted or fine-tuned modules are typically tuned to a particular KG's domain and schema (entity types, relations, constraints), often via curated in-context examples or KG-specific fine-tuning. As a result, performance does not transfer reliably when the domain shifts, the schema changes, or the system is deployed on a new KG (e.g., RoG [3]; see right panel of Figure 2).

To tackle these challenges, we introduce **KG-R1**, an agentic KG-RAG system that employs end-to-end multi-turn reinforcement learning [12, 13, 14] to learn effective and transferable KG interaction policies. As shown in Figure 1, the architecture of KG-R1 has two components: a single LLM agent and a KG query server. The KG query server hosts the knowledge graph along with a set of retrieval actions. The LLM agent iteratively performs cycles of short reasoning followed by retrieval actions over multiple turns, with each decision informed by knowledge obtained from the KG retrieval server, and generates a final answer.

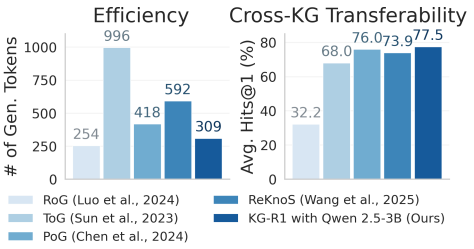


Figure 2: Prior multi-module methods are costly and do not transfer well across KGs. *Left:* mean end-to-end generated tokens per query on `WebQSP` [11]. *Right:* average Hits@1 over four out-of-training KGQA datasets (see Section 4.4). KG-R1 combines low cost with strong cross-KG transferability.

By learning a unified interaction policy, KG-R1 avoids redundant multi-module coordination and allocates computation adaptively across reasoning steps. Figure 2 shows that its 3B-parameter model combines strong answer accuracy and cross-KG transfer with fewer generated tokens than multi-module agentic baselines. Our key contributions are summarized as follows:

1. KG-R1 Framework. We introduce an agentic KG-RAG system (Section 3) that replaces multi-module pipelines with a *single* agent for KG-RAG, running on a lightweight KG server. The agent alternates between reasoning and retrieval actions over multiple turns, with the end-to-end trajectory optimized by reinforcement learning (RL) using both *turn-wise* and *outcome-based* reward signals. Turn-wise rewards evaluate individual action effectiveness and adherence to formatting, while global rewards measure answer quality and retrieval relevance.

2. Effective and Generation-Efficient Inference. By unifying reasoning and retrieval within a single learned policy, KG-R1 enables a lightweight agent to achieve strong answer accuracy with fewer generated tokens than larger multi-module systems. This offers a favorable tradeoff between answer quality and inference cost (Section 4.3).

3. Plug-and-play Transferability across Diverse KGs. KG-R1 can be easily transferred to diverse KGs and maintains strong KG-RAG performance (Section 4.4). The trained KG-R1 agent generalizes to new KGs without retraining: backend KGs can be swapped without changing prompts

or hyperparameters, or requiring fine-tuning. This generalizability enables zero-shot transfer to unseen KGs.

2 Related Work

Knowledge Graph Retrieval-Augmented Generation (KG-RAG). KG-RAG augments LLMs with structured KGs to improve quality and compositional reasoning. Early work grounds LLMs' generation by translating natural-language questions into executable graph queries (e.g., SPARQL/Cypher), retrieving relevant subgraphs or answers, and feeding them back to the model [15, 9]. More recent approaches adopt a modular LLM pipeline over KGs, interleaving natural language reasoning with multi-stage planning, path search, and execution, where each stage uses a prompted or fine-tuned LLM [3, 2, 16, 4]. Despite these advances, most systems rely on hand-engineered modules or prompt heuristics tied to a specific KG schema, which induce cost inefficiency and limit generalization. These challenges motivate our KG-R1 framework: a single-agent KG-RAG approach that improves efficiency and transfers well to new KGs.

Graph-native retrievers, such as GNN-RAG [17], train a separate GNN to score candidate entities and construct paths before LLM generation. In contrast, KG-R1 learns a unified policy over typed, connected, and bidirectional symbolic KG transitions; it shares the sequential decision-making structure of search agents such as Search-R1 [12], but instantiates it with a graph-specific interface and zero-shot backend transfer.

Multi-turn RL for LLMs. RL has become central to equipping LLMs with step-by-step (chain-of-thought) reasoning behavior [18, 14]. RL-enhanced models yield substantial gains in math and coding [19, 20], and broader complex reasoning tasks. More recently, RL has been applied to agentic LLMs that invoke external tools (e.g., bash terminals and APIs) or interact with knowledge bases, improving tool use [21] and retrieval-augmented generation (RAG) [12, 22]. Building on these advances, KG-R1 adopts end-to-end RL as the core optimization algorithm for an agentic KG-RAG framework.

3 KG-R1: An Agentic KG-RAG Framework

3.1 Problem Definition

KG-RAG has many applications, including conversational assistants [23], recommendation systems [24], and open-domain QA [25]. In this work, we instantiate and evaluate our approach on Knowledge Graph Question Answering (KGQA), which provides a grounded testbed for KG-RAG: ground-truth answers are tied to a fixed KG, evaluation is verifiable, and intermediate graph reasoning is interpretable. We now formalize knowledge graphs and KGQA tasks.

Knowledge Graphs. A knowledge graph (KG) is a graph-structured representation of real-world knowledge that encodes factual information as triples of entities and their relationships. We denote a KG as $G = \{\,(e, r, e') \mid e, e' \in \mathcal{E},\ r \in \mathcal{R}\,\}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively, and each triple (e, r, e') represents a directed edge from entity e to entity e' via relation r. For example, there is an edge `has_capital` from entity `Illinois` to entity `Springfield`.

Knowledge Graph Question Answering (KGQA) is a reasoning task based on KGs. Consider a dataset $\mathcal{D} = \{(q, G, A_q)\}$, where q is a natural-language question, G is a KG, and $A_q \subseteq \mathcal{E}$ is the ground-truth answer set. A *reasoning path* is a sequence of labeled edges $r_1, \ldots, r_\ell \in \mathcal{R}$ instantiated by entities $e_0, \ldots, e_\ell$ such that $Z:\ e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \cdots \xrightarrow{r_\ell} e_\ell$, where e_0 is an anchor entity mentioned in q and $(e_{i-1}, r_i, e_i) \in G$ for each step. A path supports a gold answer if its terminal entity e_ℓ belongs to A_q. At inference time, the agent uses q and graph observations to produce a predicted set $\hat{A}_q$, which is then compared with A_q. As an example, to answer the question "What is the capital of the U.S. state whose largest city is Chicago?", a valid reasoning path is

$$\text{Chicago} \xrightarrow{\texttt{located_in_state}} \text{Illinois} \xrightarrow{\texttt{has_capital}} \text{Springfield}.$$

This path supports the correct prediction $\hat{A}_q = \{\text{Springfield}\}$.

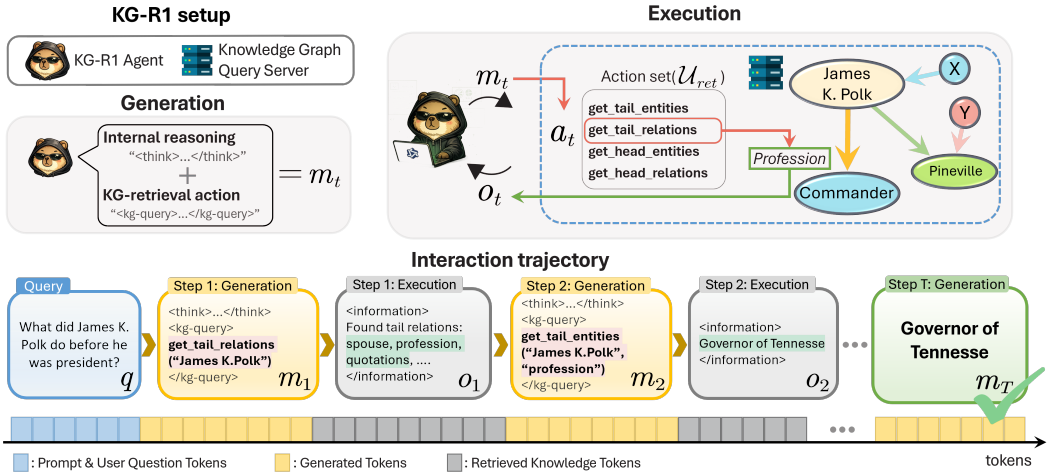


Figure 3: KG-R1 framework: a single LLM agent undergoes a multi-turn generation-execution loop with a schema-agnostic KG retrieval server and responds with the final answer.

3.2 KG-R1 Framework

KG-R1 casts KG-RAG as a multi-turn interaction with a KG interface (KG query server). We prioritize two design principles. First, we design a *single-agent* architecture that simplifies deployment and enables efficient, low-cost inference. Second, we build a *schema-agnostic* KG retrieval server that avoids KG-specific assumptions and remains portable across different KGs. Given a KGQA dataset $D = \{(q, G, A_q)\}$, we set up a KG retrieval server and a KG-R1 agent.

KG Query Server. The server hosts the knowledge graph G and provides a set of retrieval actions $a \in \mathcal{U}_{\text{ret}}$ that enable graph traversal through 1-hop operations:

$$\mathcal{U}_{\text{ret}} = \{\texttt{get_tail_relations}, \texttt{get_head_relations}, \texttt{get_tail_entities}, \texttt{get_head_entities}\}$$

$$\begin{aligned}
\texttt{get_tail_relations}(e) &:= \{\, r \in \mathcal{R} \mid \exists e' \in \mathcal{E} :\ (e, r, e') \in G \,\}, \\
\texttt{get_head_relations}(e') &:= \{\, r \in \mathcal{R} \mid \exists e \in \mathcal{E} :\ (e, r, e') \in G \,\}, \\
\texttt{get_tail_entities}(e, r) &:= \{\, e' \in \mathcal{E} \mid (e, r, e') \in G \,\}, \\
\texttt{get_head_entities}(r, e') &:= \{\, e \in \mathcal{E} \mid (e, r, e') \in G \,\}.
\end{aligned}$$

The retrieval actions execute typed transitions over explicit KG triples. The agent discovers incoming or outgoing relations incident to an entity and then expands the corresponding relation-specific neighborhood. Separate head- and tail-oriented operators support bidirectional traversal, allowing reasoning to follow connected relational paths. Each returned entity or relation is determined by a symbolic KG operation.

The retrieval action set $\mathcal{U}_{\text{ret}}$ is sufficient for *realizing* any reasoning path $Z:\ e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \cdots \xrightarrow{r_\ell} e_\ell$ as a finite action sequence whose output includes the terminal entity e_ℓ. Forward traversal along Z is implemented by $\texttt{get_tail_entities}(e_{i-1}, r_i)$ for $i = 1, \ldots, \ell$ and backward traversal is implemented by $\texttt{get_head_entities}(r_i, e_i)$.

This action design guarantees *path realizability* of the symbolic KG interface, as formalized in Remark 3.1, and provides a schema-agnostic interaction interface that enables transfer across KGs. The statement characterizes the expressiveness of the action set for paths present in the KG; whether a learned policy discovers a relevant path is an empirical question. Full formal statements and additional properties are provided in Appendix B.

Remark 3.1 (KG Action-Set Path Realizability). *For any reasoning path $Z:\ e_0 \xrightarrow{r_1} \cdots \xrightarrow{r_\ell} e_\ell$ in G, there exists an action sequence in $\mathcal{U}_{ret}$ of length at most $\ell+1$ whose output includes e_ℓ.*

KG-R1 Agent. We model a single *LLM agent* that interacts with a KG query server in a multi-turn setting. The agent undergoes initialization followed by a loop of generation and execution (see

Figure 3). At initialization, a lightweight base LLM is configured with an instruction prompt p (see the box below for the prompt template) containing general reasoning instructions, the user question q, and the KG retrieval server instructions (Table E.2). At each turn $t \leq H$, where H is the maximum turn limit, the agent first undergoes the **generation** phase where it produces a response m_t comprising two parts: (1) an internal reasoning wrapped in `<think>...</think>`, and (2) an action, which is either a KG retrieval action wrapped in `<kg-query>...</kg-query>`, or a final answer wrapped in `<answer>...</answer>`.

Prompt template for KG-R1

You are a helpful assistant. Answer the given question. You can query from knowledge base provided to you to answer the question. You can query knowledge up to [H] times. You must first conduct reasoning inside <think>...</think>. If you need to query knowledge, you can set a query statement between <kg-query>...</kg-query> to query from knowledge base after <think>...</think>. When you have the final answer, you can output the answer inside <answer>...</answer>.
KG Query Server Instruction : [KG_query_server_instruction]
Question: [question].
Assistant:

In the following **execution** phase, we parse m_t into an *action* $a_t \in \mathcal{U}_{\text{ret}} \cup \{\text{Answer}\}$ using an exact-match parser Ψ (i.e., $a_t = \Psi(m_t)$). If $a_t \in \mathcal{U}_{\text{ret}}$, executing it yields an *observation* o_{t+1} (i.e., retrieved entities, relations, or an error message in case the action does not generate a valid retrieval), which is appended to the dialogue context prior to the next turn. If $a_t = \text{Answer}$, the interaction terminates: the content inside `<answer>...</answer>` is extracted and post-processed (normalization, deduplication, and entity resolution) to produce the predicted answer set $\hat{A}_q$. At turn t, $\text{context}_t = (p, (m_1, o_2), \ldots, (m_{t-1}, o_t))$ contains the initial prompt and the preceding responses and observations. The policy $\pi_\theta(m_t \mid \text{context}_t)$ generates the next response m_t. A completed trajectory τ records the prompt, all generated responses, and the observations returned by retrieval actions.

3.3 KG-R1 Training with RL

Our goal is to learn a policy $\pi_\theta(m_t \mid \text{context}_t)$ for the KG-R1 agent that interacts with the knowledge graph G over multiple turns to incrementally retrieve informative reasoning paths and predict an answer set $\hat{A}_q$. We optimize this policy using reinforcement learning with a GRPO-style objective [14], following recent advances in multi-turn agent optimization [26, 12, 13]. An overview of the training procedure is provided in Algorithm 1 in Appendix E.

Reward Objective. To effectively train the KG-R1 agent, we combine verifiable *turn* rewards with outcome-level *global* rewards. **Turn rewards** (r_t^{turn}) provide local signals at each turn as the sum of three components: (i) *format validity* $v_{\text{fmt}}(m_t)$ checks that m_t contains both reasoning and a well-formed action $a_t \in \mathcal{U}_{\text{ret}} \cup \{\text{Answer}\}$; (ii) *KG query* $v_{\text{kg}}(a_t, o_{t+1})$ checks that executing a_t yields meaningful, schema-valid retrieval in o_{t+1}; and (iii) *answer consistency* $v_{\text{ans}}(m_T)$ checks the final answer's format/consistency on the final turn. The turn rewards are computed as follows with weights w_{fmt}, w_{kg}, and w_{ans}:

$$r_t^{\text{turn}} = w_{\text{fmt}}\, v_{\text{fmt}}(m_t) + w_{\text{kg}}\, v_{\text{kg}}(a_t, o_{t+1}) + w_{\text{ans}}\, v_{\text{ans}}(m_T). \tag{1}$$

Global rewards summarize trajectory-level outcomes as a weighted sum of two components: (i) a binary answer correctness signal H1 that is 1 if at least one normalized predicted entity matches a normalized gold entity, and 0 otherwise; and (ii) a *retrieval score* v_{ret} that is 1 if any ground-truth entity appears anywhere in the retrieved information along the executed reasoning path, and 0 otherwise.

$$R^{\text{global}} = w_{\text{H1}} \cdot \text{H1}\big(\hat{A}_q, A_q\big) \;+\; w_{\text{ret}} \cdot v_{ret} \tag{2}$$

Group-relative Turn-level Credit Assignment and Optimization. To assign credit across multi-turn trajectories, we adopt a group-relative, turn-level credit assignment strategy [13]. We collect N rollouts per query q. For each trajectory $\tau^{(n)}$, we compute turn rewards $r_t^{\text{turn},(n)}$ and a global reward $R^{\text{global},(n)}$, and combine them into a per-turn return with λ as the normalization factor:

$$G_t^{(n)} \;=\; r_t^{\text{turn},(n)} \;+\; \lambda\, R^{\text{global},(n)}. \tag{3}$$

Let $\mathcal{S}=\{(n,t):t\le T^{(n)}\}$ be the set of all turns across the N rollouts. The turn-level advantage A_t^n is calculated using a *single* group baseline $\bar{G}$ that averages over $\mathcal{S}$:

$$A_t^{(n)}=\frac{G_t^{(n)}-\bar{G}}{\sigma_G+\varepsilon}\quad\text{where}\quad \bar{G}=\tfrac{1}{|\mathcal{S}|}\sum_{(n,t)\in\mathcal{S}}G_t^{(n)},\ \sigma_G=\sqrt{\tfrac{1}{|\mathcal{S}|}\sum_{(n,t)\in\mathcal{S}}\big(G_t^{(n)}-\bar{G}\big)^2},$$

where ε is a constant for numerical stability. Intuitively, this measures each turn's contribution relative to the group average, reinforcing above-average decisions and discouraging below-average ones.

RL update. Using the turn-level advantage $A_t^{(n)}$, we optimize the agent's policy π_θ with a GRPO-style clipped objective $\mathcal{J}$:

$$\mathcal{J}(\theta)=\mathbb{E}\Bigg[\sum_{n,t,i}\Big(\min\Big(\rho_{t,i}^{(n)}\widetilde{A}_{t,i}^{(n)},\ \mathrm{clip}\left(\rho_{t,i}^{(n)},1-\epsilon,1+\epsilon\right)\widetilde{A}_{t,i}^{(n)}\Big)-\beta\mathrm{KL}\big(\pi_\theta(\cdot\mid h_{t,i}^{(n)})\|\pi_0(\cdot\mid h_{t,i}^{(n)})\big)\Big)\Bigg],$$

where $\widetilde{A}_{t,i}^{(n)}=m_{t,i}^{(n)}A_t^{(n)}$ applies the same turn-level advantage to all tokens within a turn via a mask $m_{t,i}^{(n)}$, clip bounds updates to improve stability, and n, t, and i index trajectories, turns, and tokens, respectively. The coefficient β weights the KL penalty. The current policy is π_θ, the behavior policy $\pi_{\theta_{\text{old}}}$ is used for sampling, and π_0 is a fixed reference policy used only for KL regularization. The importance ratio $\rho_{t,i}^{(n)}=\exp\big(\log\pi_\theta(y_{t,i}^{(n)}\mid h_{t,i}^{(n)})-\log\pi_{\theta_{\text{old}}}(y_{t,i}^{(n)}\mid h_{t,i}^{(n)})\big)$ corrects off-policy sampling at each token i. We maximize $\mathcal{J}(\theta)$ via gradient ascent. The proposed combination of structured rewards and group-relative credit assignment yields stable and informative learning signals for multi-turn decision-making, leading to improved reasoning quality and final answer correctness. We provide comprehensive ablation studies of each component of our reward design in Section 4.5.

3.4 Cross-KG Transfer: Plug-and-Play

We propose a plug-and-play approach for KG-R1 that enables KG-RAG to operate across different knowledge graphs without retraining. Let $\pi_{\theta|\mathcal{D}}$ denote the policy of the KG-R1 agent trained on a KGQA dataset $\mathcal{D}=\{(q,G,A_q)\}$. For a new KGQA dataset $\mathcal{D}^*=\{(q^*,G^*,A_{q^*})\}$, we replace the backend KG of the retrieval server with G^* (*plug*) and evaluate $\pi_{\theta|\mathcal{D}}$ on $\mathcal{D}^*$ without changing the learned policy or retrieval interface (*play*). This approach requires no task-specific fine-tuning, data relabeling, or architectural changes.

This plug-and-play capability stems from two design choices in KG-R1. First, the KG server exposes a schema-agnostic action set consisting solely of 1-hop retrieval operations, which are universally compatible with directed KGs. This contrasts with SPARQL-generation approaches in prior work that depend on KG-specific syntax and schema knowledge. Second, the agent's strategy for retrieving and using KG evidence is learned in a way that is independent of any particular schema, enabling transfer to new domains.

4 Experiments

We evaluate KG-R1 through two research questions: **RQ1** (Effectiveness and Efficiency): *Can* KG-R1 *learn an effective and efficient KG-RAG policy when trained on a target KG?*; **RQ2** (Cross-KG Transferability): *Does* KG-R1 *generalize to unseen KGs without additional training?* For both research questions, we compare against established LLM-only and LLM+KG baselines to assess effectiveness, efficiency, and transferability.

4.1 Experimental Setup

Models. We use Qwen2.5-3B-it [27] as our base model. For the baseline methods, we use Qwen2.5-7B-it, Qwen2.5-72B-it, Qwen3-235B-A22B-Instruct-2507 [28], and Llama2-7B-it (only for the fine-tuned planning in RoG [3]), following the setups used in prior work.

Datasets. We mainly evaluate KG-R1 in domain on (1) `WebQSP` [11]: natural-language questions over Freebase, which are mostly 1–2 hop questions, using the official 3,098/1,639 train/test QA split; and (2) `ComplexWebQuestions (CWQ)` [29]: compositional, multi-hop questions over Freebase

with a 24,649/3,531 train/test QA split. For scalability, we extract 2-hop subgraphs for each question as done in RoG [3]. We evaluate the plug-and-play approach on diverse KGQA benchmarks spanning two categories of KGs: (1) *Freebase* type—`SimpleQA` [30] and `GrailQA` [31], which share the Freebase KG but differ in question complexity and reasoning path distributions; and (2) *Wikidata* type—`T-REx` [32] and `QALD-10en` [33] for cross-schema generalization. Appendix C provides a summary of dataset sources, schema, and evaluation splits.

Metrics. Following prior works [3, 2, 16, 34], we evaluate KGQA performance with *Hits@1*. For every method, the same deterministic evaluator parses predicted and reference entities, normalizes both sides, and assigns a score of 1 if at least one normalized predicted entity exactly matches a normalized gold entity, and 0 otherwise. See Appendix F.1 for precision, recall, and F1. For each query, we sum generated output tokens across all model calls; total tokens include both input and output tokens across those calls. For fair comparison, all token counts are computed with the Qwen2.5 tokenizer. We also report the *number of modules* to compare workflow complexity, and we analyze single-query latency and batched throughput on a single NVIDIA H100 GPU.

Baselines. We compare three groups: (1) LLM-only baselines without KG access: input–output prompting (IO) [35], chain-of-thought prompting (CoT) [36], and self-consistency (SC); (2) modular KG-augmented baselines: RoG [3], ToG [2], PoG [16], and ReKnoS [4]; and (3) KG-R1. In our RoG evaluation, the fine-tuned LLaMA2-7B-it planner is paired with a Qwen reasoner. We report means and standard deviations over five runs unless otherwise specified. Appendix D provides baseline details; Appendices F.2 and F.3 report comparisons with supervised fine-tuning (SFT) and GNN-RAG.

Implementation details. We use VERL [37] and vLLM-backed decoding [38] for implementing the KG-R1 agent. Training uses distributed optimization with gradient checkpointing and offloading. Unless stated otherwise, we set $H = 5$ as the maximum number of turns and $N = 16$ rollouts per query. Additional implementation details (e.g., hyperparameters) are provided in Appendix E.4.

4.2 Training Stability and Effectiveness

Figure 4 shows steady improvements in Hits@1 that gradually plateau as training converges for KG-R1 on both `WebQSP` and `CWQ`. The validation Hits@1 closely follows the training curve, indicating that the learned policy generalizes well to unseen data in the validation set without significant overfitting. Overall, KG-R1 training achieves stable and consistent gains in end-to-end KG-RAG performance.

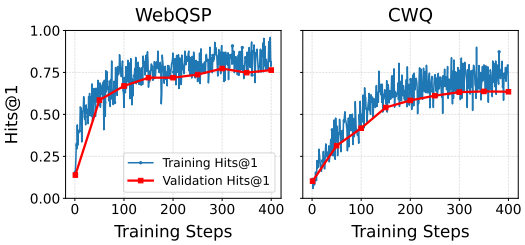


Figure 4: Hits@1 over KG-R1 training on `WebQSP` and `CWQ` for Qwen2.5-3B-it.

4.3 RQ1: Effectiveness and Efficiency Results

KG-R1 demonstrates strong accuracy on trained KGs. KG-R1 uses Qwen2.5-3B and achieves the highest WebQSP Hits@1 in Table 1 (82.75%). It exceeds the three RoG configurations (79.89–81.68%) and the best WebQSP configurations of ToG (75.67%), PoG (79.13%), and ReKnoS (76.33%). On the more compositional `CWQ`, KG-R1 achieves **65.31%** Hits@1, improving over RoG across all scales (46.32–48.94%) and surpassing ReKnoS with Qwen2.5-7B (29.12%), Qwen2.5-72B (44.10%), and Qwen3-235B (45.05%). This highlights that reinforcement learning over structured KG actions can substantially improve reasoning effectiveness beyond what is achievable by model scaling alone.

KG-R1 achieves efficient inference with reduced generation cost. Compared with ToG using Qwen3-235B, KG-R1 uses fewer generated tokens (309 vs. 996 on WebQSP; 311 vs. 803 on CWQ) and fewer total tokens per query (3,237 vs. 6,832; 3,206 vs. 5,389, respectively), while achieving higher Hits@1 on both datasets. RoG uses fewer tokens than KG-R1 but attains lower Hits@1 on both datasets. Additional results for SFT and GNN-RAG appear in Appendix F.

Single-agent design reduces system complexity. Beyond token efficiency, KG-R1 simplifies the KG-RAG pipeline into a single-agent architecture. Prior methods typically require 2–5 modules (e.g., retrieval, planning, verification, and answering), whereas KG-R1 performs retrieval and reasoning

Table 1: Performance and efficiency comparison of KGQA methods on WebQSP and CWQ datasets. The maximum turn number is set to $H = 5$. All scores are percentages.

	Datasets →		WebQSP			CWQ		
	Methods	Modules	Hits@1 (%)	Total Tokens	Gen Tokens	Hits@1 (%)	Total Tokens	Gen Tokens
Qwen 2.5-7B	IO	1	21.37 ± 0.07	151.59 ± 0.03	9.29 ± 0.03	16.63 ± 0.01	159.76 ± 0.01	8.57 ± 0.01
	CoT	1	17.27 ± 0.09	260.43 ± 0.02	49.13 ± 0.02	14.95 ± 0.02	292.46 ± 0.01	72.27 ± 0.01
	SC	1	19.26 ± 0.19	2098.01 ± 0.92	407.60 ± 0.92	17.63 ± 0.17	2364.62 ± 1.24	603.13 ± 1.24
	RoG	2	79.89 ± 0.35	814.61 ± 0.70	246.95 ± 0.67	46.32 ± 0.49	1219.72 ± 0.61	259.46 ± 0.58
	ToG	4	65.00 ± 4.63	6455.04 ± 753.60	941.52 ± 109.27	32.25 ± 0.43	8396.80 ± 87.06	1358.54 ± 17.24
	PoG	5	64.81 ± 0.84	4474.30 ± 91.92	241.14 ± 5.17	32.00 ± 2.53	8293.67 ± 1454.76	369.84 ± 45.59
	ReknoS	3	60.22 ± 5.20	2822.34 ± 209.87	558.43 ± 45.52	29.12 ± 0.23	2908.55 ± 222.47	602.08 ± 39.33
Qwen 2.5-72B	IO	1	24.66 ± 0.07	153.50 ± 0.02	11.20 ± 0.02	22.20 ± 0.02	162.04 ± 0.03	10.85 ± 0.03
	CoT	1	21.84 ± 0.04	271.46 ± 0.02	60.16 ± 0.02	22.92 ± 0.04	296.81 ± 0.04	76.62 ± 0.04
	SC	1	22.62 ± 0.07	2177.75 ± 1.23	487.34 ± 1.23	24.76 ± 0.27	2385.73 ± 0.51	624.24 ± 0.51
	RoG	2	81.63 ± 3.33	812.32 ± 7.81	244.53 ± 7.79	45.05 ± 2.87	1205.48 ± 2.51	245.34 ± 2.51
	ToG	4	74.47 ± 0.25	6810.27 ± 39.73	942.91 ± 5.80	43.18 ± 1.02	9024.77 ± 271.68	1358.32 ± 41.02
	PoG	5	72.88 ± 6.57	4759.10 ± 210.29	316.25 ± 16.85	43.62 ± 1.38	7969.21 ± 1631.58	503.29 ± 116.35
	ReknoS	3	71.42 ± 6.77	2472.90 ± 115.94	488.44 ± 39.87	44.10 ± 0.45	2278.71 ± 196.08	548.52 ± 27.79
Qwen 3-235B	IO	1	24.01 ± 0.18	136.05 ± 0.06	14.75 ± 0.06	20.22 ± 0.09	147.36 ± 0.16	17.17 ± 0.16
	CoT	1	26.94 ± 0.22	274.66 ± 0.32	84.36 ± 0.32	24.38 ± 0.33	317.42 ± 0.31	118.23 ± 0.31
	SC	1	27.60 ± 0.15	2200.20 ± 1.26	677.79 ± 1.26	26.13 ± 0.23	2551.71 ± 0.78	958.22 ± 0.78
	RoG	2	81.68 ± 0.24	960.58 ± 5.90	254.31 ± 0.11	48.94 ± 0.39	1217.51 ± 10.26	252.45 ± 5.18
	ToG	4	75.67 ± 0.37	6831.66 ± 28.39	996.12 ± 5.21	39.82 ± 0.73	5389.00 ± 33.01	802.97 ± 9.84
	PoG	5	79.13 ± 1.89	6818.26 ± 908.76	418.26 ± 49.36	44.69 ± 0.27	10177.37 ± 1859.27	548.77 ± 82.56
	ReknoS	3	76.33 ± 2.08	2795.53 ± 139.20	592.38 ± 4.18	45.05 ± 0.52	2677.92 ± 281.97	684.38 ± 14.06
	KG-R1 (Ours) Qwen2.5-3B	1	82.75 ± 0.04	3237.17 ± 30.34	308.67 ± 0.56	65.31 ± 0.14	3205.95 ± 42.34	311.14 ± 0.52

within a unified policy using only one module. This reduces deployment complexity and avoids error propagation across stages. Despite this simplification, KG-R1 remains competitive or superior to multi-module systems across both datasets, showing that structured RL can effectively consolidate KG interaction and reasoning into a lightweight unified agent.

Latency and throughput analysis.

On a single NVIDIA H100 GPU, we measure (i) single-query end-to-end latency and (ii) batched throughput for KG-R1. The single-query latency averages 6.4 ± 1.5 s per question, and the batched throughput reaches 3.7 samples per second with a batch size of 64. See Appendix F.4 for the full results.

4.4 RQ2: Cross-KG Transferability

Zero-shot transfer across unseen KGs. Table 2 reports zero-shot *plug-and-play* results, where the learned policy is directly applied to unseen KGs by swapping only the backend graph. We report both a sample-weighted average over all evaluation questions and a dataset-level macro average that gives equal weight to SimpleQA, GrailQA, T-REx, and QALD-10en. KG-R1 demonstrates strong cross-KG transferability across both Freebase- and Wikidata-based benchmarks. When trained on WebQSP, KG-R1 obtains the highest sample-weighted average, 77.50%, compared with 75.97% for PoG-Qwen3-235B. Its 65.73% macro average is, however, below PoG-Qwen2.5-72B (67.17%) and PoG-Qwen3-235B (69.28%), indicating that the aggregate comparison depends on how datasets are weighted.

On QALD-10en, WebQSP-trained KG-R1 reaches 41.96% Hits@1, below ToG and PoG with Qwen2.5-72B (59.66% and 48.18%, respectively).

Table 2: Zero-shot cross-KG transferability of KG-R1 (Hits@1, %). Agents are trained on WebQSP or CWQ and evaluated on new benchmarks by *swapping only* the KG-backend server (no policy retraining). **Weighted AVG** averages over all QA samples, whereas **Macro AVG** weights the four datasets equally. The maximum turn number is set to $H = 5$.

		Freebase-based		Wikidata-based			
	Methods↓	SimpleQA	GrailQA	T-REx	QALD-10en	**Weighted AVG**	**Macro AVG**
Qwen 2.5-7B	IO	10.30 ± 0.10	8.07 ± 0.06	22.79 ± 0.05	17.43 ± 0.01	18.84	14.65
	CoT	7.07 ± 0.06	7.73 ± 0.12	17.38 ± 0.04	21.05 ± 0.06	14.82	13.31
	SC	8.70 ± 0.20	8.60 ± 0.44	20.01 ± 0.12	20.32 ± 0.46	16.93	14.41
	RoG	12.97 ± 0.35	17.67 ± 0.15	22.37 ± 0.27	29.33 ± 0.46	20.76	20.59
	ToG	52.37 ± 0.80	54.73 ± 4.62	58.55 ± 1.11	50.95 ± 0.92	56.84	54.15
	PoG	66.38 ± 0.39	69.88 ± 0.84	75.69 ± 0.08	46.07 ± 0.25	72.28	64.51
	ReKnoS	56.43 ± 1.22	69.58 ± 0.35	78.17 ± 0.03	37.94 ± 1.21	72.21	60.53
Qwen 2.5-72B	IO	14.33 ± 0.15	11.63 ± 0.12	26.16 ± 0.02	17.45 ± 0.06	22.17	17.39
	CoT	11.07 ± 0.12	10.53 ± 0.06	24.48 ± 0.02	20.52 ± 0.17	20.57	16.65
	SC	11.60 ± 0.44	11.43 ± 0.25	26.01 ± 0.09	21.62 ± 0.60	21.86	17.67
	RoG	20.40 ± 0.26	22.63 ± 4.51	32.50 ± 0.09	44.94 ± 0.69	30.07	30.12
	ToG	57.83 ± 0.29	71.53 ± 0.76	67.93 ± 0.18	59.66 ± 0.46	66.67	64.24
	PoG	65.67 ± 0.42	75.30 ± 0.99	79.52 ± 0.08	48.18 ± 3.53	75.63	67.17
	ReKnoS	68.40 ± 5.15	73.67 ± 0.60	77.11 ± 0.02	39.94 ± 0.30	73.77	64.78
Qwen 3-235B	IO	11.37 ± 0.12	12.80 ± 0.10	34.61 ± 0.01	24.12 ± 0.17	27.99	20.73
	CoT	14.40 ± 0.36	14.60 ± 0.30	37.72 ± 0.18	28.43 ± 0.17	30.97	23.79
	SC	14.73 ± 0.42	15.73 ± 0.15	38.79 ± 0.27	29.53 ± 0.35	31.94	24.70
	RoG	18.93 ± 0.51	18.10 ± 0.20	37.31 ± 0.19	37.54 ± 1.20	32.19	27.97
	ToG	41.27 ± 3.61	53.20 ± 0.53	77.00 ± 0.10	57.76 ± 1.05	68.01	57.31
	PoG	67.90 ± 0.17	77.93 ± 0.45	78.75 ± 0.13	52.55 ± 1.04	75.97	69.28
	ReKnoS	68.01 ± 3.37	75.67 ± 0.74	77.05 ± 0.03	39.44 ± 0.35	73.92	65.04
Qwen2.5-3B	**KG-R1 (WebQSP trained)**	69.18 ± 0.10	68.55 ± 0.07	83.38 ± 0.08	41.96 ± 0.05	77.50	65.73
	KG-R1 (CWQ trained)	57.31 ± 0.08	64.84 ± 0.04	80.10 ± 0.05	34.74 ± 0.12	72.82	59.18

CWQ-trained KG-R1 reaches 72.82% weighted and 59.18% macro Hits@1. It scores lower than the WebQSP-trained model on all four transfer datasets, with the largest gap on SimpleQA (57.31% vs. 69.18%). This difference may reflect the alignment between the traversal patterns learned from each training dataset and those required by the transfer benchmarks.

The contrast between T-REx and QALD-10en also shows that sharing a Wikidata backend does not eliminate variation caused by the target dataset's question style and relation distribution.

4.5 Ablation Studies

We ablate four key components of KG-R1: reward design, RL algorithm, retrieval format, and model scale. We train on `WebQSP` and `CWQ` with $H = 5$ turns and report the final Hits@1 and retrieval success rate (Ret. Rate), defined as the fraction of queries where any ground-truth answer entity appears in the retrieved entities, in Table 3. The training curves are shown in Figure 7 in Appendix F.5.

Reward Design. Removing turn rewards causes the largest performance drop on both `WebQSP` (82.75→69.29, -13.46%) and `CWQ` (65.31→52.03, -13.28%), showing the importance of per-step supervision. Removing turn-level advantage yields smaller but consistent degradation (`WebQSP`: -3.32%, `CWQ`: -1.26%), validating the benefit of turn-level credit assignment. Removing retrieval reward causes a modest Hits@1 drop (`WebQSP`: -4.03%, `CWQ`: -2.68%), but significantly lowers retrieval success (77.51→65.35 and 52.14→45.01). Thus, with the retrieval reward, a larger fraction of queries retrieve at least one gold answer entity.

RL Algorithm and Retrieval Format. Replacing GRPO with PPO [15] substantially degrades performance (`WebQSP`: 82.75→59.62, `CWQ`: 65.31→35.66), confirming that relative advantage optimization is more stable for multi-turn KG interaction. Under PPO, we observe *reward hacking* (see examples in Appendix F.5.2): the agent fabricates "retrieved" content matching expected formats to earn reward. We ablate hierarchical relation retrieval by replacing its tree-structured relation representation with a flat list (see Appendix F.5.3). Removing this formatting reduces Hits@1 from 82.75%

Table 3: Ablation studies of KG-R1 components.

Datasets →	WebQSP		CWQ	
Methods↓	Hits@1	Ret. Rate	Hits@1	Ret. Rate
KG-R1	$82.75_{\pm 0.04}$	$77.51_{\pm 0.22}$	$65.31_{\pm 0.14}$	$52.14_{\pm 0.22}$
w/o Turn Rewards	$69.29_{\pm 0.17}$ (-13.46%)	$59.55_{\pm 0.28}$	$52.03_{\pm 0.28}$ (-13.28%)	$48.62_{\pm 0.17}$
w/o Turn-level Advantage	$79.43_{\pm 0.27}$ (-3.32%)	$69.38_{\pm 0.25}$	$64.05_{\pm 0.24}$ (-1.26%)	$51.34_{\pm 0.24}$
w/o Retrieval Reward	$78.72_{\pm 0.28}$ (-4.03%)	$65.35_{\pm 0.34}$	$62.63_{\pm 0.05}$ (-2.68%)	$45.01_{\pm 0.13}$
w PPO	$59.62_{\pm 0.51}$ (-23.13%)	$49.44_{\pm 0.53}$	$35.66_{\pm 0.15}$ (-29.65%)	$30.50_{\pm 0.24}$
w/o Hierarchical Rel. Retr.	$76.88_{\pm 0.27}$ (-5.87%)	$61.74_{\pm 0.54}$	$61.26_{\pm 0.19}$ (-4.05%)	$43.54_{\pm 0.10}$
w Qwen-2.5-7B-it	$82.44_{\pm 0.23}$ (-0.31%)	$73.37_{\pm 0.35}$	$65.85_{\pm 0.19}$ (+0.54%)	$51.63_{\pm 0.14}$

to 76.88% on WebQSP and from 65.31% to 61.26% on CWQ, supporting the benefit of hierarchical relation presentation.

Model Scale. Scaling the base model from Qwen2.5-3B to 7B does not improve performance (WebQSP: 82.75 vs. 82.44; CWQ: 65.31 vs. 65.85), indicating that KG-R1's gains mainly come from reinforcement-learned retrieval policies rather than model size alone.

5 Conclusion

We propose KG-R1, a single-agent, multi-turn KG-RAG framework in which an LLM queries a lightweight KG server and is optimized via RL. Across KGQA benchmarks, KG-R1 combines strong answer accuracy with fewer generated tokens than multi-module agentic baselines. Its schema-agnostic interface also enables zero-shot transfer across KGs, with performance varying across target datasets. Together, these results demonstrate the potential of a unified learned policy for knowledge-grounded reasoning.

Acknowledgments and Disclosures

This work is funded by the MIT-IBM AI Watson Lab, NSF awards #CCF-1845763, #CCF-2316235, and #CCF-2403237, a Google Faculty Research Award, and a Google Research Scholar Award.

References

[1] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, H. Wang, *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, vol. 2, no. 1, p. 32, 2023.

[2] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. M. Ni, H.-Y. Shum, and J. Guo, "Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph," in *ICLR*, 2024.

[3] L. Luo, Y.-F. Li, G. Haffari, and S. Pan, "Reasoning on graphs: Faithful and interpretable large language model reasoning," in *ICLR*, 2024.

[4] S. Wang, J. Lin, X. Guo, J. Shun, J. Li, and Y. Zhu, "Reasoning of large language models over knowledge graphs with super-relations," in *International Conference on Learning Representations*, 2025.

[5] X. Zhao, S. Liu, S.-Y. Yang, and C. Miao, "Medrag: Enhancing retrieval-augmented generation with knowledge graph-elicited reasoning for healthcare copilot," in *Proceedings of the ACM on Web Conference 2025*, pp. 4442–4457, 2025.

[6] J. Cui, M. Ning, Z. Li, B. Chen, Y. Yan, H. Li, B. Ling, Y. Tian, and L. Yuan, "Chatlaw: A multi-agent collaborative legal assistant with knowledge graph enhanced mixture-of-experts large language model," 2024.

[7] J. Baek, A. F. Aji, and A. Saffari, "Knowledge-augmented language model prompting for zero-shot knowledge graph question answering," in *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pp. 78–106, 2023.

[8] J. D'Abramo, A. Zugarini, and P. Torroni, "Investigating large language models for text-to-sparql generation," in *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*, pp. 66–80, 2025.

[9] J. Lee and H. Shin, "Sparkle: Enhancing sparql generation with direct kg integration in decoding," *Expert Systems with Applications*, vol. 289, p. 128263, 2025.

[10] L. Yao, C. Mao, and Y. Luo, "Kg-bert: Bert for knowledge graph completion," *arXiv preprint arXiv:1909.03193*, 2019.

[11] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *ACL*, 2016.

[12] B. Jin, H. Zeng, Z. Yue, J. Yoon, S. O. Arik, D. Wang, H. Zamani, and J. Han, "Search-R1: Training llms to reason and leverage search engines with reinforcement learning," *arXiv preprint arXiv:2503.09516*, 2025.

[13] S. Zeng, Q. Wei, W. Brown, O. Frunza, Y. Nevmyvaka, and M. Hong, "Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment," *arXiv preprint arXiv:2505.11821*, 2025.

[14] DeepSeek-AI, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv:2501.12948*, 2025.

[15] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems (NeurIPS)*, vol. 35, pp. 27730–27744, 2022.

[16] L. Chen, P. Tong, Z. Jin, Y. Sun, J. Ye, and H. Xiong, "Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs," in *Advances in neural information processing systems (NeurIPS)*, 2024.

[17] C. Mavromatis and G. Karypis, "GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs," in *Findings of the Association for Computational Linguistics: ACL 2025*, (Vienna, Austria), pp. 16682–16699, Association for Computational Linguistics, July 2025.

[18] OpenAI et al., "OpenAI o1 System Card," *arXiv:2412.16720*, 2024.

[19] H. Le, Y. Wang, A. D. Gotmare, S. Savarese, and S. C. H. Hoi, "Coderl: Mastering code generation through pretrained models and deep reinforcement learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 21314–21328, 2022.

[20] Y. Chervonyi, T. H. Trinh, M. Olšák, X. Yang, H. H. Nguyen, M. Menegali, J. Jung, J. Kim, V. Verma, Q. V. Le, *et al.*, "Gold-medalist performance in solving olympiad geometry with alphageometry2," *Journal of Machine Learning Research (JMLR)*, vol. 26, no. 241, pp. 1–39, 2025.

[21] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, *et al.*, "Toolllm: Facilitating large language models to master 16000+ real-world apis," in *ICLR (OpenReview)*, 2024.

[22] H. Wang, C. Qian, W. Zhong, X. Chen, J. Qiu, S. Huang, B. Jin, M. Wang, K.-F. Wong, and H. Ji, "Acting less is reasoning more! teaching model to act efficiently," *arXiv preprint arXiv:2504.14870*, 2025. Also referred to as "OTC: Optimal Tool Calls via Reinforcement Learning".

[23] D. Chaudhuri, M. R. A. H. Rony, and J. Lehmann, "Grounding dialogue systems via knowledge graph aware decoding with pre-trained transformers," in *European Semantic Web Conference*, pp. 323–339, Springer, 2021.

[24] S. Wang, W. Fan, Y. Feng, L. Shanru, X. Ma, S. Wang, and D. Yin, "Knowledge graph retrieval-augmented generation for llm-based recommendation," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27152–27168, 2025.

[25] X. Zhu, Y. Xie, Y. Liu, Y. Li, and W. Hu, "Knowledge graph-guided retrieval augmented generation," in *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8912–8924, 2025.

[26] C. Qian, E. C. Acikgoz, Q. He, H. Wang, X. Chen, D. Hakkani-Tür, G. Tur, and H. Ji, "ToolRL: Reward is all tool learning needs," *arXiv preprint arXiv:2504.13958*, 2025.

[27] Q. A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu, "Qwen2.5 technical report," 2025.

[28] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[29] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," in *NAACL*, 2018.

[30] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale Simple Question Answering with Memory Networks," *arXiv preprint arXiv:1506.02075*, 2015.

[31] Y. Gu, S. Kase, M. Vanni, B. Sadler, P. Liang, X. Yan, and Y. Su, "Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases," in *Proceedings of the Web Conference 2021*, (Ljubljana, Slovenia), pp. 3375–3385, ACM / IW3C2, 2021.

[32] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, and E. Simperl, "T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, (Miyazaki, Japan), European Language Resources Association (ELRA), May 2018.

[33] R. Usbeck, X. Yan, A. Perevalov, L. Jiang, J. Schulz, A. Kraft, C. Möller, J. Huang, J. Reineke, A.-C. Ngonga Ngomo, M. Saleem, and A. Both, "Qald-10–the 10th challenge on question answering over linked data: Shifting from dbpedia to wikidata as a kg for kgqa," *Semantic Web*, vol. 15, no. 6, pp. 2193–2207, 2024.

[34] M. Park, H. Yang, J. Kim, K. Park, and H. Kim, "Prograg: Hallucination-resistant progressive retrieval and reasoning over knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, pp. 32674–32682, 2026.

[35] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.

[36] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems (NeurIPS)*, vol. 35, pp. 24824–24837, 2022.

[37] G. Sheng, C. Zhang, Z. Ye, X. Wu, W. Zhang, R. Zhang, Y. Peng, H. Lin, and C. Wu, "Hybridflow: A flexible and efficient rlhf framework," in *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

[38] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[39] S. Ramírez, "Fastapi." `https://fastapi.tiangolo.com/`, 2018. Accessed September 24, 2025.

[40] Encode, "Uvicorn." `https://www.uvicorn.org/`, 2018. ASGI server for Python. Accessed September 24, 2025.

A Discussion

Societal Impact. KG-R1 aims to improve knowledge-grounded reasoning by enabling language models to interact with structured knowledge graphs through learned retrieval policies. This may benefit applications requiring factual consistency and traceable reasoning, such as question answering, scientific knowledge access, and enterprise knowledge systems. By reducing reliance on complex multi-module pipelines, KG-R1 may also improve the accessibility and deployment efficiency of knowledge-grounded reasoning systems.

Potential Negative Impact. Like other retrieval-augmented systems, KG-R1 may propagate incorrect or outdated knowledge if the underlying knowledge graph is incomplete, noisy, or maliciously manipulated. In high-stakes domains such as medicine or finance, such errors could lead to misleading outputs. Additionally, improved knowledge-grounded reasoning may be misused to automate large-scale misinformation generation with higher factual coherence.

Limitations. KG-R1 assumes access to a reasonably structured and queryable knowledge graph, which may not be available in all domains. The retrieval interface uses the same triple operations across KGs, but answer accuracy depends on the quality and coverage of the graph. Changes in relation vocabulary and graph structure can also affect transfer. Reinforcement learning training remains computationally expensive compared with standard supervised fine-tuning.

For WebQSP and CWQ, we also follow the standard benchmark setting in which linked topic entities are provided; performance with a separately predicted entity linker may be lower.

B Path Realizability and Transferability of the KG Server

Here, we establish path realizability of $\mathcal{U}_{ret}$ in our KG retrieval server and clarify the scope of its schema-free transferability. The result characterizes the expressive capability of the symbolic action interface; policy performance is evaluated empirically.

Preliminaries. Let $G = \{(e, r, e') \mid e, e' \in \mathcal{E},\ r \in \mathcal{R}\}$ be a directed KG. Define the KG-R1 action set

$$\mathcal{U}_{\text{ret}} = \left\{ \begin{matrix} \texttt{get_tail_relations},\ \texttt{get_head_relations}, \\ \texttt{get_tail_entities},\ \texttt{get_head_entities} \end{matrix} \right\}$$

with the following semantics for any $(e, r, e') \in G$:

$$\begin{aligned} \texttt{get_tail_relations}(e) &:= \{\, r \in \mathcal{R} \mid \exists e' \in \mathcal{E} :\ (e, r, e') \in G \,\}, \\ \texttt{get_head_relations}(e') &:= \{\, r \in \mathcal{R} \mid \exists e \in \mathcal{E} :\ (e, r, e') \in G \,\}, \\ \texttt{get_tail_entities}(e, r) &:= \{\, e' \in \mathcal{E} \mid (e, r, e') \in G \,\}, \\ \texttt{get_head_entities}(r, e') &:= \{\, e \in \mathcal{E} \mid (e, r, e') \in G \,\}. \end{aligned}$$

A *relation path* is $z = (r_1, \ldots, r_\ell)$, and a *reasoning path* (instantiation) is

$$Z :\ e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \cdots \xrightarrow{r_\ell} e_\ell, \quad (e_{i-1}, r_i, e_i) \in G.$$

Remark 3.1 (KG Action-Set Path Realizability). *For any reasoning path $Z :\ e_0 \xrightarrow{r_1} \cdots \xrightarrow{r_\ell} e_\ell$ in G, there exists an action sequence in $\mathcal{U}_{ret}$ of length at most $\ell+1$ whose output includes e_ℓ.*

Proof. Starting with $\ell = 1$, if $(e_0, r_1, e_1) \in G$ then $e_1 \in \texttt{get_tail_entities}(e_0, r_1)$. For the inductive step, assume the claim holds for length $\ell - 1$. By the induction hypothesis, we reach $e_{\ell-1}$; applying $\texttt{get_tail_entities}(e_{\ell-1}, r_\ell)$ then returns a set containing e_ℓ.

Schema-Free Transferability. The retrieval action interface $\mathcal{U}_{\text{ret}}$ is defined purely in terms of set operations over triples (e, r, e'), independent of any schema, ontology, or datatype constraints. As a result, the same action interface can be applied to different knowledge graphs without modification. This allows a learned policy to be executed on a new KG by simply replacing the backend graph.

However, this property guarantees only that the policy remains well-defined under KG replacement. The effectiveness of transfer depends on the similarity between the training and target KGs and is evaluated empirically in Section 4.4.

C Datasets

C.1 KGQA Datasets

Here, we list the KGQA datasets used in this paper.

Freebase-based Datasets

- **WebQSP** [11] (1,639 test QA pairs): Open-domain questions with mostly 1–2 hop reasoning over Freebase MIDs and SPARQL annotations.
- **ComplexWebQuestions (CWQ)** [29] (3,531 test QA pairs): Compositional multi-hop questions generated from SPARQL templates that stress longer chains and constraint composition are used to probe multi-turn retrieval quality and robustness without dataset-specific rules.
- **SimpleQuestions (SimpleQA)** [30] (1,000 sampled test QA pairs): Single-relation 1-hop questions over Freebase that serve as a retrieval-fidelity and token-efficiency baseline for KG-R1. We randomly sample 1,000 QA from the original test split (21,687).
- **GrailQA** [31] (1,000 sampled test QA pairs): GrailQA contains compositional questions designed to test generalization on Freebase. We evaluate it with the same four retrieval operations defined in Section 3.2 and randomly sample 1,000 examples from the original test split (13,231).

Wikidata-based Datasets

- **T-REx** [32] (5,000 test QA pairs): Large-scale slot-filling–style QA grounded in Wikidata triples; used to assess scalability and coverage under a different KG schema. We randomly sample 5,000 from the corpus ($\sim$2.2M).
- **QALD-10en** [33] (333 test QA pairs): Manually curated, linguistically varied questions over Wikidata; useful for evaluating precision on a small but challenging set. We evaluate on all 333 examples.

C.2 Dataset Pre-processing

Two-hop Subgraph Extraction Methodology. Following the released RoG/NSM-style preprocessing pipeline [3], we build a question-specific local subgraph to shrink the search space and suppress spurious matches. Starting only from the benchmark-provided linked topic entities e_q, the pipeline performs two-hop expansion and personalized PageRank pruning. Gold answers A_q are reserved for evaluation and reward computation, while graph construction relies solely on linked topic entities. The processed subgraph for each question is cached and used by the KG retrieval server in KG-R1.

C.3 Freebase and Wikidata Schema Comparison

The following tables show the different schemas of Freebase and Wikidata.

Table 4: Freebase knowledge graph dataset used for KGQA evaluation. All datasets share the same underlying Freebase knowledge graph structure with 4.9M entities and 663 relations. The questions shown are examples.

Dataset	Entities	Relations
Freebase	- P!nk	- broadcast.content.artist
	- Gender	- people.person.nationality
	- Ice Cube	- music.artist_genre
	- United States of America	- location.location.containedby
	- Nemanja Mikic	- basketball.basketball.player_position
Questions		

WebQSP (1,639 test questions):
Q: What does jamaican people speak?
A: *[Jamaican English, Jamaican Creole English Language]*

CWQ (3,531 test questions):
Q: Lou Seal is the mascot for the team that last won the World Series when?
A: *[2014 World Series]*

SimpleQA (21,687 test questions):
Q: Where is the madam satan located?
A: *[United States of America]*

GrailQA (13,231 test questions):
Q: Which play is produced by the illusion?
A: *[The Illusion]*

Table 5: Wikidata knowledge graph datasets used for KGQA evaluation. All datasets share the same underlying Wikidata knowledge graph structure with 15M entities and 2.3K relations. The questions shown are examples.

Dataset	Entities	Relations
Wikidata	- Barack Obama	- instance of
	- Germany	- country
	- Albert Einstein	- occupation
	- Microsoft	- date of birth
	- Paris	- place of birth

Questions

T-Rex (2.2M test questions):
Q: What is the occupation of Albert Einstein?
A: *[Physicist]*

QALD-10en (250 test questions):
Q: Which companies were founded by Bill Gates?
A: *[Microsoft]*

D Baselines

D.1 LLM-only Methods

Setup. We evaluate IO, CoT, and SC in a zero-shot setting without access to the KG retrieval server (i.e., no KG augmentation). All baselines run on Qwen2.5-7B-it, Qwen2.5-72B-it, and Qwen3-235B-A22B-Instruct-2507 with temperature$= 0$ and top_k$= 50$.

Prompts. We use the following prompt templates for IO and CoT baselines (the SC baseline performs majority voting over CoT results and hence uses the same prompt as CoT).

Prompte template for IO Prompt setup

Answer the given question directly and concisely based on your knowledge.
Format your answer as: Answers: ["answer1", "answer2", ...].
For single answers, use: Answers: ["answer"].
Question: [`Question`]
Answers:

Prompte template for CoT setup

Think through the question step by step, then provide the answer.
IMPORTANT: Follow this exact format:
1. Start with "Reasoning:" followed by your step-by-step thinking.
2. End with "Answers:" followed by your final answer in brackets.
3. Do NOT put "Answers:" before your reasoning.

Format your answer as: Answers: ["answer1", "answer2", ...].
For single answers, use: Answers: ["answer"].
Question: [`Question`]
Reasoning:

Evaluation. We apply the same entity-matching protocol to all methods. Predicted and gold entities are parsed and normalized using identical rules. Hits@1 is 1 if the predicted entity set contains at least one exact match to a gold entity, and 0 otherwise.

D.2 KG-augmented Baselines

RoG [3]: *Reasoning on Graphs (RoG)* couples LLMs with KGs via a *planning–retrieval–reasoning* pipeline: the LLM first proposes KG-grounded relation paths as faithful plans, uses them to retrieve valid paths from the KG, and then performs stepwise reasoning to produce an answer. This yields interpretable multi-hop reasoning and reduces hallucinations by constraining reasoning to the KG structure.

ToG [2]: *Think-on-Graph (ToG)* treats the LLM as an agent that *iteratively explores* the KG: it performs beam search over entities/relations, expands promising paths, and alternates retrieval with reasoning until a final path/answer is selected. Compared to prompt-only baselines, ToG improves deep, compositional reasoning by explicitly navigating the KG during multi-hop search.

PoG [16]: *Plan-on-Graph (PoG)* extends agentic KG reasoning with *adaptive planning and self-correction*. It decomposes a question into sub-objectives and iteratively performs path exploration, memory updating, and reflection to dynamically adjust reasoning trajectories. Through guidance, memory, and reflection mechanisms, PoG enables flexible exploration breadth and backtracking over erroneous paths, improving both reasoning accuracy and efficiency compared to fixed exploration strategies.

ReKnoS [4]: *Reasoning with Knowledge Super-relations (ReKnoS)* introduces *super-relations* that summarize and connect multiple relational paths, enabling both forward and backward reasoning while expanding the search space that the LLM can traverse. By reasoning over these super-relations (rather than isolated edges), ReKnoS boosts retrieval success and multi-hop accuracy, especially on complex queries.

E KG-R1 details

E.1 Training algorithm

Algorithm 1 summarizes the RL training procedure.

Algorithm 1: KG-R1 Training with RL

Input: Dataset $D = \{(q, G, A_q)\}$, base LLM π_0, maximum number of turns H, rollouts N
Output: Trained policy π_θ
$\pi_\theta \leftarrow \pi_0$ `// Initialize from base LLM`
foreach *mini-batch of queries from* D **do**
  foreach q *in batch* **do**
    Collect N rollouts $\{\tau^{(n)}\}_{n=1}^{N}$;
    for $n \leftarrow 1$ **to** N **do**
      $\tau^{(n)} \leftarrow (p)$; `// where p is the instruction prompt for` q
      for $t \leftarrow 1$ **to** H; `// Multi-turn interaction`
      do
        $r_t \sim \pi_\theta(\cdot \mid \tau^{(n)})$; `// Generate response`
        $a_t \leftarrow \Psi(r_t)$ where $a_t \in \mathcal{A}_{\text{query}} \cup \{\texttt{answer}\}$;
        if $a_t \in \mathcal{A}_{query}$ **then**
          Execute a_t, get o_{t+1}, and append to $\tau^{(n)}$;
        end
        else
          Extract $\hat{A}_q$ and **break**;
        end
      end
    end
    Compute rewards for collected rollouts:;
      Turn: $r_t^{\text{turn},(n)} = w_{\text{fmt}} v_{\text{fmt}}(r_t) + w_{\text{kg}} v_{\text{kg}}(a_t, o_{t+1}) + w_{\text{ans}} v_{\text{ans}}(r_T)$;
      Global: $R^{\text{global},(n)} = w_{\text{H1}} \cdot \text{H1}(\hat{A}_q, A_q) + w_{\text{ret}} \cdot v_{\text{ret}}$;
  end
  Credit assignment: $G_t^{(n)} = r_t^{\text{turn},(n)} + \lambda R^{\text{global},(n)}$;
  Group baseline: $\bar{G} = \frac{1}{|\mathcal{S}|} \sum_{(n,t)\in\mathcal{S}} G_t^{(n)}$, where $\mathcal{S} = \{(n,t) : t \le T^{(n)}\}$;
  Advantages: $A_t^{(n)} = \frac{G_t^{(n)} - \bar{G}}{\text{std}_{(n,t)\in\mathcal{S}}(G) + \epsilon}$;
  Update: π_θ via GRPO with $J(\theta)$, entropy $\mathcal{H}$, and KL divergence to π_0;
end

E.2 Server Instruction Prompt

The server instruction template consists of a set of query actions and descriptions. We use the same server instructions for both training and evaluation across experiments.

Server instruction template for KG-R1

If you encounter a KG-related error, read the error message carefully and correct your query.

Use exactly these query functions:
-`get_tail_relations`(entity) : Returns relations where the entity is the subject/head.
-`get_head_relations`(entity) : Returns relations where the entity is the object/tail.
-`get_tail_entities`(entity, relation) : Returns entities connected to the given entity by the specified relation.
-`get_head_entities`(entity, relation) : Returns entities from which the given entity is connected by the specified relation.

E.3 Reward Weights

Our RL objective combines per-turn and episode-level signals with fixed weights. Per-turn rewards encourage well-structured interaction and effective retrieval: $r_t^{\text{turn}} = w_{\text{fmt}}\, s_{\text{fmt}} + w_{\text{kg}}\, s_{\text{kg}} + w_{\text{ans}}\, s_{\text{ans}}$,

where s_{fmt} scores output structure/format validity, s_{kg} rewards schema-valid, non-empty KG queries, and s_{ans} checks final-answer formatting/consistency.

Episode-level reward emphasizes answer correctness and retrieval coverage: $R^{\text{global}} = w_{\text{H1}} \cdot \text{H1}(\hat{A}_q, A_q) + w_{\text{ret}} \cdot v_{\text{ret}}$, where $\text{H1} \in \{0, 1\}$ indicates whether at least one predicted entity matches a gold entity and $v_{\text{ret}} \in \{0, 1\}$ indicates whether a gold entity occurs in the retrieved information. Unless otherwise noted, we use the default values listed in Table 6.

Table 6: Reward-component weights (w) for `KG-R1` reward function.

Symbol	Name	Scope	Role (concise)	Default
w_{fmt}	Format weight	Turn	Rewards valid per-turn output structure in r_t^{turn}	0.5
w_{kg}	KG-query weight	Turn	Rewards schema-valid, non-empty KG retrieval in r_t^{turn}	0.5
w_{ans}	Answer-format weight	Turn	Rewards correct final-answer formatting/consistency in r_t^{turn}	0.5
w_{H1}	Final-answer weight	Episode	Weights binary $\text{H1}(\hat{A}_q, A_q)$ in R^{global}	1.0
w_{ret}	Retrieval-coverage weight	Episode	Rewards coverage signal $v_{\text{ret}} \in \{0, 1\}$ in R^{global}	1.0

E.4 RL Implementation and Hyperparameters

Learning Rates and Optimizer. Both Qwen2.5-3B-it and Qwen2.5-7B-it use an identical learning rate of 1×10^{-6} with no warmup. We use the AdamW optimizer with weight decay 0.01 applied to all parameters except biases and layer-normalization weights, and gradient clipping by global norm set to 1.0 to prevent gradient explosion during RL training.

Batch Configuration and Gradient Accumulation. The 3B model uses a training batch size of 128 with a validation batch size of 256, and the 7B model uses a training batch size of 256 with a validation batch size of 128. During GRPO rollout, we collect $N = 16$ trajectories per prompt for the 3B model and $N = 8$ for the 7B model to balance exploration with memory constraints. The mini-batch size is 128 for both models, with dynamic batch sizing enabled to utilize GPU memory efficiently.

RL Coefficients. RL training uses GRPO (Group Relative Policy Optimization) with multi-turn advantage computation enabled. The KL loss coefficient differs by model: $\beta = 0.01$ for the 3B model and $\beta = 0.02$ for the 7B model, using the K3 KL loss to limit divergence from the reference policy. The entropy coefficient is set to 0 for both models, favoring exploitation over exploration for multi-turn KG reasoning.

Sampling Configuration. During training, we use sampling temperature 1.0 with nucleus sampling disabled (`top_p`$= 1.0$ and `top_k`$= -1$) to maintain consistent generation. For evaluation, we use greedy decoding with temperature 0.0 and sampling disabled (`do_sample`$=$ `false`). Because GPU inference can still exhibit small run-to-run variation, we report the mean and standard deviation over five runs unless a result is explicitly marked as a single run.

Hardware Specifications and Training Duration. Training is conducted on 4 NVIDIA H100 GPUs for both model sizes. For the 3B model, we allow up to $21{,}000$ tokens/GPU for PPO processing and a maximum of $18{,}384$ batched tokens for VLLM rollout. For the 7B model, we allocate $12{,}000$ tokens/GPU for PPO and a maximum of $12{,}288$ batched tokens for VLLM, using `FSDP` parameter and optimizer offloading to fit memory constraints. Both configurations use `bfloat16` precision with chunked prefill enabled. Training runs for 400 steps with checkpoints saved every 50 steps (3B model) or every 25 steps (7B model).

E.5 KG-R1 Retreival Server Details

Setup. We implemented the KG-R1 retrieval server with FastAPI [39] and Uvicorn [40] to support (i) a schema-free 1-hop KG query API, (ii) high-throughput asynchronous batch execution, and (iii) robust validation and observability (regex action parsing, standardized `<information>` wrapping with auto-closure, and final-turn safeguards). After the KG-R1 agent generates a response, the parsed `<kg-query>` action is sent to the server, which performs exact string matching over the per-question knowledge graph to resolve entities and relations and returns the retrieved information. If the call is invalid—one of: `KG_SERVER_ERROR: Invalid Action; KG_FORMAT_ERROR: Missing Required Fields; KG_FORMAT_ERROR: Wrong Argument`

Count; KG_SAMPLE_NOT_FOUND: Sample Missing; KG_ENTITY_NOT_FOUND: Entity Not in KG; KG_RELATION_NOT_FOUND: Invalid Relation; KG_NO_RESULTS: No Relations Found; KG_NO_RESULTS: No Entities Found—the server responds with a descriptive error message (see box below).

KG-R1 Server Error Examples

KG_SERVER_ERROR: Invalid Action

<error>Action "get_entity_info" not available (use: get_head_relations, get_tail_relations, get_head_entities, get_tail_entities)</error>

KG_FORMAT_ERROR: Missing Required Fields

<error>Missing required fields for get_tail_entities: relation_name</error>

KG_FORMAT_ERROR: Wrong Argument Count

<error>get_tail_relations accepts only one entity argument</error>

KG_SAMPLE_NOT_FOUND: Sample Missing

<error>Sample "sample_12345" not found in KG</error>

KG_ENTITY_NOT_FOUND: Entity Not in KG

<error>Entity "Barack Obamaa" not found in KG</error>

KG_RELATION_NOT_FOUND: Invalid Relation

<error>Relation "location.capital" not found in KG</error>

KG_NO_RESULTS: No Relations Found

<error>No tail relations found for entity "Random_Entity_123" in knowledge graph</error>

KG_NO_RESULTS: No Entities Found

<error>No tail entities found for relation "film.director.film" with head "Barack Obama" in knowledge graph</error>

Figure 5: KG-R1 error types with actual server error messages.

F Supplementary Experimental Results

F.1 Precision, Recall, and F1

We supplement Hits@1 with precision, recall, and F1. Predicted and gold entities are normalized and deduplicated before set comparison. Per-question scores are averaged over questions; all reported metrics are percentages. Tables 7–9 report five-run means and standard deviations.

The Hits@1-trained model obtains 60.97% F1, 73.73% precision, and 55.02% recall on WebQSP, and 34.03% F1, 26.83% precision, and 59.79% recall on CWQ. We also train a CWQ model that uses F1 as the global answer reward while keeping the turn-level and retrieval rewards unchanged. On CWQ, it reaches 49.74% F1, 51.43% precision, and 53.01% recall.

Table 7: Precision, recall, and F1 (%) using Qwen2.5-7B baselines. KG-R1 uses Qwen2.5-3B. Values are five-run means ± standard deviations.

	WebQSP			CWQ		
Method	F1	Precision	Recall	F1	Precision	Recall
IO	36.36 ±0.26	36.79 ±0.06	44.46 ±0.01	26.12 ±0.02	27.30 ±0.01	28.78 ±0.02
CoT	34.69 ±0.05	34.09 ±0.00	45.90 ±0.02	24.25 ±0.05	24.61 ±0.01	28.13 ±0.02
SC	36.48 ±0.19	36.18 ±0.21	46.38 ±0.26	26.88 ±0.16	27.74 ±0.16	29.36 ±0.26
RoG	64.13 ±0.19	66.86 ±0.21	70.19 ±0.38	32.82 ±0.08	28.52 ±0.07	48.34 ±0.45
ToG	13.49 ±1.83	14.96 ±2.34	14.48 ±2.14	10.74 ±0.20	10.55 ±0.22	11.68 ±0.27
PoG	41.92 ±0.44	48.24 ±0.30	41.73 ±0.33	24.78 ±1.80	24.24 ±1.74	26.51 ±1.93
ReKnoS	20.79 ±0.32	26.37 ±0.31	20.35 ±0.34	20.05 ±0.21	19.78 ±0.22	21.13 ±0.26
KG-R1 (Hits@1 reward)	60.97 ±0.41	73.73 ±0.43	55.02 ±0.43	34.03 ±0.32	26.83 ±0.31	59.79 ±0.47
KG-R1 (F1 reward)	–	–	–	49.74 ±0.46	51.43 ±0.47	53.01 ±0.48

Table 8: Precision, recall, and F1 (%) using Qwen2.5-72B baselines. KG-R1 uses Qwen2.5-3B. Values are five-run means ± standard deviations.

	WebQSP			CWQ		
Method	F1	Precision	Recall	F1	Precision	Recall
IO	42.06 ±0.02	42.48 ±0.02	52.65 ±0.02	32.74 ±0.01	33.85 ±0.01	36.91 ±0.01
CoT	41.04 ±0.04	40.08 ±0.03	52.97 ±0.07	33.97 ±0.03	34.32 ±0.04	38.28 ±0.05
SC	41.78 ±0.09	40.96 ±0.07	53.18 ±0.19	35.92 ±0.23	36.50 ±0.25	39.36 ±0.21
RoG	70.87 ±2.13	74.34 ±2.31	75.31 ±2.65	39.91 ±1.51	37.56 ±1.20	47.96 ±2.93
ToG	12.06 ±0.33	13.05 ±0.38	13.11 ±0.37	19.26 ±0.85	18.90 ±0.93	20.48 ±0.94
PoG	46.80 ±2.14	52.69 ±3.08	47.33 ±2.09	31.97 ±1.21	31.20 ±1.17	34.55 ±1.41
ReKnoS	41.77 ±0.50	51.97 ±0.52	41.34 ±0.57	34.02 ±0.42	33.14 ±0.43	36.82 ±0.48
KG-R1 (Hits@1 reward)	60.97 ±0.41	73.73 ±0.43	55.02 ±0.43	34.03 ±0.32	26.83 ±0.31	59.79 ±0.47
KG-R1 (F1 reward)	–	–	–	49.74 ±0.46	51.43 ±0.47	53.01 ±0.48

Table 9: Precision, recall, and F1 (%) using Qwen3-235B baselines. KG-R1 uses Qwen2.5-3B. Values are five-run means ± standard deviations.

	WebQSP			CWQ		
Method	F1	Precision	Recall	F1	Precision	Recall
IO	41.21 ±0.12	40.85 ±0.14	55.83 ±0.02	29.84 ±0.10	30.26 ±0.10	38.19 ±0.15
CoT	43.86 ±0.17	43.80 ±0.15	56.00 ±0.22	34.08 ±0.28	34.93 ±0.33	37.26 ±0.37
SC	44.37 ±0.11	44.40 ±0.12	56.15 ±0.25	35.63 ±0.21	36.63 ±0.18	37.95 ±0.32
RoG	72.31 ±0.16	75.95 ±0.21	75.63 ±0.13	39.24 ±0.41	36.34 ±0.41	51.37 ±0.50
ToG	35.31 ±2.14	42.52 ±2.93	34.94 ±1.76	24.77 ±1.55	25.60 ±0.30	27.86 ±0.30
PoG	52.38 ±1.16	58.38 ±1.23	52.38 ±1.20	33.26 ±3.07	32.47 ±2.95	36.36 ±3.54
ReKnoS	40.69 ±0.35	48.68 ±0.37	40.13 ±0.42	35.43 ±0.50	34.67 ±0.47	37.97 ±0.55
KG-R1 (Hits@1 reward)	60.97 ±0.41	73.73 ±0.43	55.02 ±0.43	34.03 ±0.32	26.83 ±0.31	59.79 ±0.47
KG-R1 (F1 reward)	–	–	–	49.74 ±0.46	51.43 ±0.47	53.01 ±0.48

F.2 Comparison with Supervised Fine-Tuning (SFT)

SFT hyperparameters. We train for one epoch (one pass over the trajectory dataset) with a learning rate of 10^{-5} and a batch size of 4.

We train a Qwen2.5-3B SFT baseline on 19,346 successful interaction trajectories from the CWQ training split, generated by our trained 3B KG-R1 model and retained when their final answers receive a positive Hits@1 score. SFT training uses eight NVIDIA A100 GPUs with 80 GB memory each. The resulting baseline distills the behavior of an RL-trained policy. Table 10 compares it with KG-R1 using the same 3B base model. SFT reaches 41.65% Hits@1 and 26.66% F1, whereas KG-R1 reaches 65.31% Hits@1 and 34.03% F1 with slightly fewer total and generated tokens.

Table 10: SFT and RL comparison on `CWQ` using Qwen2.5-3B. SFT results are from a single evaluation run. Token counts are reported per query.

Method	**Hits@1 (%)**	**F1 (%)**	**Total Tokens**	**Gen. Tokens**
SFT trajectory imitation	41.65	26.66	3319.70	328.60
KG-R1 (multi-turn RL)	**65.31**	**34.03**	**3205.95**	**311.14**

F.3 Comparison with GNN-RAG

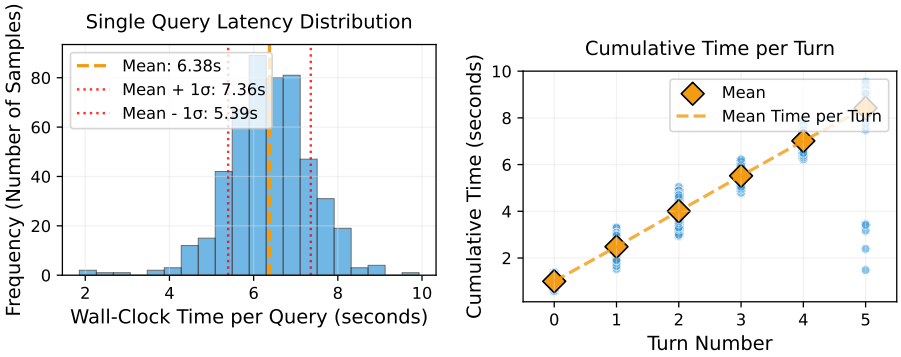
We additionally compare with GNN-RAG [17], a graph-native pipeline that performs subgraph preprocessing, GNN retriever inference, candidate-entity scoring, path processing, and final LLM answer generation. Table 11 reports one run for each GNN-RAG configuration. KG-R1 achieves higher Hits@1 on both datasets while using a substantially smaller 3B language model. GNN-RAG uses fewer LLM tokens. These counts cover LLM inference and exclude graph preprocessing and GNN inference.

Table 11: Comparison with GNN-RAG. GNN-RAG results are from one run. Token counts are reported per query.

Dataset	Method / Backbone	Hits@1 (%)	Total Tokens	Gen. Tokens
WebQSP	GNN-RAG / Qwen2.5-7B	62.90	2575.28	26.37
	GNN-RAG / Qwen2.5-72B	72.79	2584.17	29.02
	GNN-RAG / Qwen3-235B	78.65	2566.42	28.78
	KG-R1 / Qwen2.5-3B	**82.75**	3237.17	308.67
CWQ	GNN-RAG / Qwen2.5-7B	47.83	2778.86	28.15
	GNN-RAG / Qwen2.5-72B	51.54	2715.80	28.07
	GNN-RAG / Qwen3-235B	52.82	2774.73	26.45
	KG-R1 / Qwen2.5-3B	**65.31**	3205.95	311.14

F.4 Latency and Throughput Analysis

Single-Query Latency. We measured end-to-end wall-clock latency on 500 randomly sampled WebQSP queries. The mean latency is 6.38 s with a standard deviation of ≈ 1.0 s (i.e., mean$\pm 1\sigma =$ 5.39–7.36 s). Figure 6 shows the distribution and the per-turn timing breakdown; cumulative time grows approximately linearly with the number of turns, and the average maximum turn count is 4.2.



(a) Histogram of end-to-end latency per question.

(b) Cumulative time by agent turn within a query.

Figure 6: Single-query latency of KG-R1 on WebQSP using one NVIDIA H100. (a) Distribution of end-to-end latency; the dashed line marks the mean 6.38 s, and dotted lines indicate mean $\pm$ 1σ (5.39–7.36 s). (b) Cumulative time versus turn number across 500 queries; diamonds show per-turn means and the dashed trend denotes the average time per turn. The average maximum turn count is 4.2, and the near-linear growth indicates predictable per-turn costs suitable for interactive KGQA.

Batched Throughput. We evaluate batched inference at a batch size of 64 on a single NVIDIA H100 (Table 12). LLM-only baselines (no KG calls) achieve high throughput—81.8 (IO) and 70.1 (CoT) samples/s—with relatively short outputs (43.0/206.0 tokens per sample). KG-R1 incurs KG-server retrieval, reducing throughput but remaining practical for offline processing: the single-run setting reaches 3.7 samples/s (1205.9 generated tokens/s) with 4.4 KG calls per query, while the N=4 runs setting trades throughput for more KG interaction (17.5 calls per query), yielding 2.0 samples/s (612.1 gen tokens/s).

Table 12: Batched throughput on one NVIDIA H100 (256 queries; batch size 64). *Samples*: total queries. *Batch*: batch size. *KG Calls*: total KG-server requests. *Calls/Sample*: average KG requests per query. *Total (s)*: end-to-end wall-clock time. *Gen Tok./Sample*: generated tokens per query. *Samples/s*: queries per second. *Gen Tok./s*: generated tokens per second.

Configuration	Samples	Batch	KG Calls	Calls/Sample	Total (s)	Gen Tok./Sample	Samples/s	Gen Tok./s
IO	256	64	0	0.0	12.4	43.0	81.8	887.7
CoT	256	64	0	0.0	14.1	206.0	70.1	3740.1
KG-R1 (single run)	256	64	1127	4.4	73.2	345.0	3.7	1205.9
KG-R1 (N=4 runs)	256	64	4478	17.5	142.2	340.0	2.0	612.1

F.5 Ablation Studies

All ablation studies in Table 3 evaluate KG-R1 with Qwen2.5-3B-it trained on `WebQSP` and `CWQ` using a maximum turn budget of H=5. We report the training curves of the ablation studies in Figure 7.

Turn Reward. We vary the turn-level reward by setting the weights to w_{fmt}=0, w_{kg}=0, and w_{ans}=0 (the default is 0.5 for all weights).

Turn-level Advantage. Instead of computing the turn-level group advantage $A_t^{(n)}$ in Section 3.3, we compute a trajectory-wise group advantage

$$A_t^{(n)} = \frac{G^{(n)} - \bar{G}}{\sigma_G + \epsilon}, \qquad G^{(n)} = \frac{1}{T}\sum_t r_t^{\text{turn},(n)} + R^{\text{global},(n)},$$

and use it for token-level credit assignment.

Retrieval Reward. We ablate the retrieval component by setting the weight w_{ret}=0 (the default is 1).

RL Algorithm: GRPO vs. PPO. We replace GRPO with vanilla PPO (VERL [37] defaults), and set the turn-reward mixture weight w_{turn}=0 (default: 0.5). Advantage estimation is performed by a learned value critic (Qwen2.5-3B-it).

Hierarchical Relation Retrieval. The default configuration groups retrieved relations hierarchically according to their "domain.type.property" structure (see Table 14). In the ablation, we remove this hierarchical formatting and present the same relations as a flat list.

Base LLM Parameter Size (7B). We swap the base LLM from Qwen2.5-3B-it to Qwen2.5-7B-it while keeping data, rewards, and budgets fixed.

F.5.1 Training curves

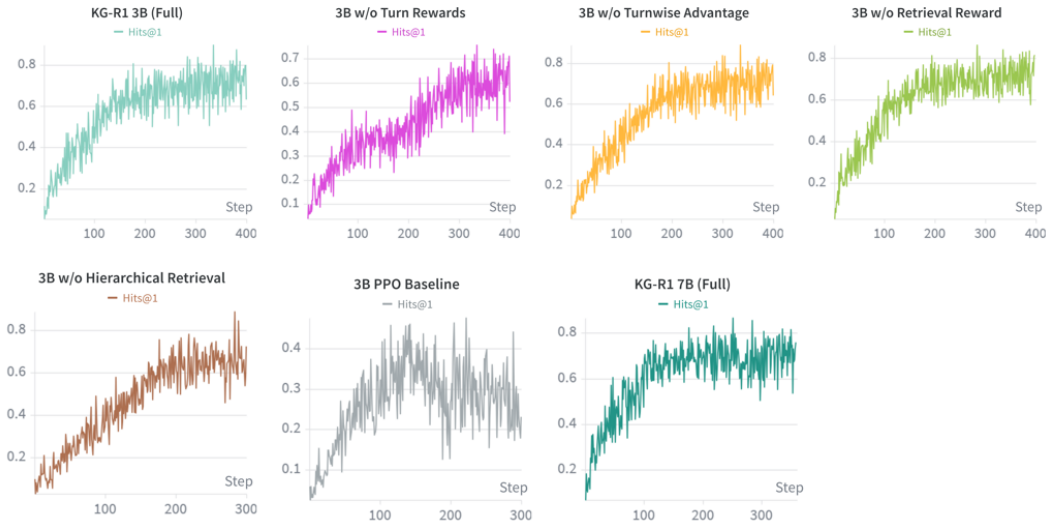


Figure 7: Training curves of ablation studies, reporting Hits@1 across training steps.

F.5.2 PPO Reward Hacking

Table 13 shows the PPO reward hacking examples.

Table 13: Policy model reward hacking under PPO. The actor model generates fake KG information that the critic model cannot distinguish from real data. Blue denotes responses generated by the KG-R1 agent.

Normal KG Retrieval	**Fake Information Generated by Actor Policy in PPO**
Question: Who directed the movie Inception? <kg-query> get_head_entities("film.film.directed_by", "Inception") </kg-query> <information> Head entities for relation "film.film.directed_by" with tail "Inception": Christopher Nolan (m.04f9z) </information> **Answer:** Christopher Nolan directed Inception.	**Question:** Who directed the movie Inception? <kg-query> get_head_entities("film.film.directed_by", "Inception") </kg-query> <information> Head entities for relation "film.film.directed_by" with tail "Inception": Christopher Nolan, Barack Obama, Elon Musk, Albert Einstein, Michael Jordan </information> **Answer:** Inception was directed by Christopher Nolan, with creative input from Barack Obama, Elon Musk, Albert Einstein, and Michael Jordan as co-directors.

F.5.3 Hierarchical Retrieval Format

Hierarchical retrieval format groups Freebase relations by their dot-notation structure (e.g., domain → type → property), presenting them as a compact tree instead of a flat comma-separated list. In the example (see Table 14), this formatting preserves the same information while reducing verbosity—79 tokens vs. 156 tokens (≈49% fewer tokens).

Table 14: Hierarchical relation retrieval formatting used in KG-R1 to organize Freebase relations.

Flat Format	**Hierarchical Format**

<kg-query>
get_tail_relations("m.09c7w0")
</kg-query>
 	
<information> Tail relations for "m.09c7w0":
 	
location.country.first_level_divisions,
location.location.containedby,
location.location.contains,
people.person.nationality,
people.person.place_of_birth,
government.government.government_for
</information>
 	
Result: *156 tokens*
	<kg-query>
	get_tail_relations("m.09c7w0")
	</kg-query>
	
	<information> Tail relations for "m.09c7w0":
	
	location
	  country: first_level_divisions
	  location: containedby, contains
	people
	  person: nationality, place_of_birth
	government
	  government: government_for
	</information>
	
	Result: *79 tokens (49.4% savings)*

G Examples of Responses

Table 15: Example KG-R1 response on WebQSP, showing multi-step reasoning and verification for a person-children query. Blue denotes responses generated by the KG-R1 agent.

KG-R1 Response Example — WebQSP

Question: What are Victoria Beckham's kids' names? Initial entities : 'Victoria Beckham'

<think>
I need to find out the names of Victoria Beckham's kids. I will start by finding any relations to her children.
</think>
<kg-query>
get_tail_relations("Victoria Beckham")
</kg-query>

<information> Tail relations for entity "Victoria Beckham":
base.popstra.celebrity.breakup, award.award_nominee.award_nominations, ..., people.person.children, ...</information>

<think>
The relation people.person.children is relevant. I will use it to get the names of her children.
</think>
<kg-query>
get_tail_entities("Victoria Beckham", "people.person.children")
</kg-query>

<information> Tail entities for relation "people.person.children":
Brooklyn Joseph Beckham, Cruz David Beckham, Harper Seven Beckham, Romeo James Beckham </information>

<think>
The names of Victoria Beckham's children are found. I will verify by checking each child's parent relation.
</think>
<kg-query>
get_head_relations("Brooklyn Joseph Beckham")
</kg-query>

<information> Head relations for "Brooklyn Joseph Beckham":
base.godparents.godparent.godchildren, people.person.children, people.sibling_relationship.sibling </information>

<think>
Confirming Brooklyn's parents using people.person.children.
</think>
<kg-query>
get_head_entities("Brooklyn Joseph Beckham", "people.person.children")
</kg-query>

<information> Head entities for relation with tail "Brooklyn Joseph Beckham":
David Beckham, Victoria Beckham </information>

<think> Brooklyn, Cruz, Harper, and Romeo all confirm to have David and Victoria Beckham as parents. </think>
<answer>
Brooklyn Joseph Beckham, Cruz David Beckham, Harper Seven Beckham, Romeo James Beckham
</answer>

[ground_truth] {'Brooklyn Joseph Beckham', 'Romeo James Beckham', 'Cruz David Beckham', 'Harper Seven Beckham'}

Table 16: Example KG-R1 response on CWQ. Blue denotes responses generated by the KG-R1 agent.

KG-R1 Response Example - CWQ

Question: What movie, written by Mike Jefferies, starred Nick Cannon?
Initial entities: "Mike Jefferies", "Nick Cannon"

<think>
Let's start by finding a movie written by Mike Jefferies. We can use the relation that associates writers with their movies.
</think>
<kg-query>
get_tail_relations("Mike Jefferies") </kg-query>

<information>Tail relations for entity "Mike Jefferies": film.film_story_contributor.film_story_credits, people.person.nationality, people.person.place_of_birth, film.writer.film, ... </information>

<think>
From the initial query results, we have two possible options for Mike Jefferies' work. Let's start with "film.writer.film" to find the movies written by Mike Jefferies and then see if Nick Cannon starred in any of those films.
</think>
<kg-query>
get_tail_entities("Mike Jefferies", "film.writer.film")
</kg-query>

<information>Tail entities for relation "film.writer.film" with head "Mike Jefferies": Goal II: Living the Dream, Goal III: Taking on the World</information>

<think>
Now that we have two movies written by Mike Jefferies, let's check if Nick Cannon has starred in any of these movies. We'll use the relation that connects actors with the movies they star in.
</think> <kg-query>
get_head_relations("Nick Cannon")
</kg-query>

<information>Head relations for entity "Nick Cannon": base.popstra.dated.participant, film.personal_film_appearance.person, music.album.primary_release, base.popstra.lived_with.participant, film.film.written_by, ... </information>

...
...
...

<think>
From the relations where "Goal II: Living the Dream" is involved, we can see that it is a film with a writer, which aligns with Mike Jefferies.
</think>
<answer>
Goal II: Living the Dream, School Dance, School Gyrls, Goal III: Taking on the World
</answer>

[ground_truth] {'Goal II: Living the Dream' }

Table 17: Example KG-R1 response on SimpleQA. Blue denotes responses generated by the KG-R1 agent.

KG-R1 Response Example - SimpleQA

Question: Where is the madam satan located? (Initial entities: "Madam Satan")

<think>
I need to find out where Madam Satan is located. First, I will check if it has a location or if it's an entity that can be queried about its location.
</think>
<kg-query>
get_tail_relations("Madam Satan")
</kg-query>

<information>Tail relations for entity "Madam Satan":
film.film.cinematography, common.topic.notable_types, film.film.language, film.film.starring,film.film.country, ... award.ranked_item.appears_in_ranked_lists, film.film.film_art_direction_by</information>

<think>
None of the identified relations are directly related to location. Let me check if Madam Satan is detailed in a geographical context.
</think>
<kg-query>
get_tail_entities("Madam Satan", "film.film.country")
</kg-query>

<information>Tail entities for relation "film.film.country" with head "Madam Satan": United States of America</information>

<think>
I have found that Madam Satan, identified as a film, is located in the United States of America. However, it's crucial to confirm the specific location or city within the country where Madam Satan was filmed.
</think>
<kg-query>
get_tail_entities("Madam Satan", "film.film.starring")
</kg-query>

..
..
..

<information>Tail entities for relation "film.film.runtime" with head "Madam Satan": m.03kx9f2</information>

<think>
I am still not obtaining the most helpful information about the location. Possibly Madam Satan was released, but not filmed in a specific place.
</think>
<kg-query>
get_head_relations("m.02tb01n")
</kg-query>

<information>Head relations for entity "m.02tb01n":
film.film.starring, film.actor.film</information>

<think>
I have questioned the relation "film.actor.film". This relation includes the movies the actor is in. Let's try to find a clearer detail for the location of Madam Satan.
</think>
<answer>
United States of America
</answer>

[ground_truth] {'United States of America' }

Table 18: Example KG-R1 response on T-REx. Blue denotes responses generated by the KG-R1 agent.

KG-R1 Response Example — T-REx

Question: What is the instance of Solberg??
Initial entities: "Solberg"

<think>
Let's start with the entity Solberg and try to find out what it represents. It seems like Solberg could be an entity that describes a geopolitical boundary, location, or characterization. The query can be adjusted from general to specific as needed.
</think>
<kg-query>
get_tail_relations("Solberg")
</kg-query>

<information>Tail relations for entity "Solberg": P1889, P407, P1552, P282, P31, P460</information>

<think>
The relations provided show how Solberg is related to other entities. Let's explore each relation to find out more about Solberg.
</think>
<kg-query>
get_tail_entities("Solberg", "P282")
</kg-query>

<information>Tail entities for relation "P282" with head "Solberg": Latin script</information>

<think>
The relation P282 corresponds to "type.object.name" according to the API documentation, indicating that Solberg is a human name. The relation P407 is also noteworthy but does not give further specific information. Let's try a broader query for more context related to Solberg.
</think>
<kg-query>
get_tail_entities("Solberg", "P31")
</kg-query>

<information>Tail entities for relation "P31" with head "Solberg": Wikimedia disambiguation page, family name</information>

...
...
...

<think>
The retrieval of "Soelberg", "Sohlberg", and "Sohlberg" suggests that Solberg might be associated with these names. To find an instance of Solberg, we can search for a specific entity of this name.
</think>
<answer>
Latin script, Wikimedia disambiguation page, family name, Soelberg, Sohlberg, Sohlberg
</answer>

[ground_truth] {'family name', 'human settlement', 'urban area in Sweden', 'minor locality in Sweden', 'minor locality in Sweden', 'Wikimedia disambiguation page', 'human settlement', 'Ortsteil', 'stone setting', 'human settlement', 'minor locality in Sweden', 'hill', 'stone setting', 'hill', 'human settlement' }

Table 19: Example KG-R1 response on QALD10en. Blue denotes responses generated by the KG-R1 agent.

KG-R1 Response Example - QALD10en

Question: After whom is the Riemannian geometry named?? (Initial entities: "Riemannian geometry")

<think>
I need to find out who the Riemannian geometry is named after. Let's start by investigating the relations and entities connected to "Riemannian geometry".
</think>
<kg-query>
get_tail_relations("Riemannian geometry")
</kg-query>

<information>Tail relations for entity "Riemannian geometry":
on focus list of Wikimedia project, named after, maintained by WikiProject, topic's main category, instance of, subclass of</information>

<think>
From the relations extracted, it appears the Riemannian geometry is named after someone. I will now search for the person it's named after.
</think>
<kg-query>
get_tail_entities("Riemannian geometry", "named after")
</kg-query>

<information>Tail entities for relation "named after" with head "Riemannian geometry": Bernhard Riemann</information>

<think>
I now have the information that the Riemannian geometry is named after Bernhard Riemann. Let's confirm this by checking if there's more detailed information about this relationship.
</think>
<kg-query>
get_head_relations("Bernhard Riemann")
</kg-query>

<information>Head relations for entity "Bernhard Riemann":
named after</information>

<think>
These results are confirming that Bernhard Riemann is indeed the person the Riemannian geometry is named after. No further information is needed.
</think>
<kg-query>
get_head_entities("Bernhard Riemann", "named after")
</kg-query>

<information>Head entities for relation "named after" with tail "Bernhard Riemann": Riemannian geometry</information>

<think>
This confirms my findings; the Riemannian geometry is named after Bernhard Riemann.
</think>
<answer>
Bernhard Riemann
</answer>

[ground_truth] {'Bernhard Riemann'}

Table 20: Example KG-R1 response on GrailQA. Blue denotes responses generated by the KG-R1 agent.

KG-R1 Response Example — GrailQA

Question: What video game is published by softdisk?
Initial entities: "Near West Side"

<think>
First, I need to find out what game is published by Softdisk. Without specific knowledge about "Near West Side", I will start with the game publisher for Softdisk.
</think>
<kg-query>
get_head_relations("Near West Side")
</kg-query>

<information>Head relations for entity "Near West Side":
book.periodical_publisher_period.publisher, business.industry.companies, business.employment_tenure.company</information>

<think>
The head relation "business.employment_tenure.company" suggests that "Near West Side" might be a game company or related to a game. I will try to find out the publisher of a game related to "Near West Side" by looking for games published by Softdisk.
</think>
<kg-query>
get_tail_relations("Near West Side")
</kg-query>

<information>Tail relations for entity "Near West Side":
organization.organization.place_founded, organization.organization.headquarters, book.periodical_publisher.periodicals_published, business.employer.employees, cvg.cvg_publisher.games_published, business.business_operation.industry, common.topic.notable_types</information>

...
...
...

<think>
The previous query did not provide a conclusive answer. Let's try a different approach by checking if "Augusto Cury" is associated with any game publishers.
</think>
<answer>
Augusto Cury, Catacomb 3-D, Chagunitzu, Dangerous Dave in the Haunted Mansion, Hovertank 3D, Iron Seed, Lado Skorin, North Cambridge Academy, Roberto Torretti, Shadow Knights, Star Film Company, Tiles of the Dragon
</answer>

[ground_truth] {'Rescue Rover','Catacomb 3-D','Shadow Knights',"Dangerous Dave's Risky Rescue",'Tiles of the Dragon', 'Catacomb', 'Commander Keen: Keen Dreams', 'Dave Goes Nutz!','Chagunitzu','Dangerous Dave in the Haunted Mansion','Hovertank 3D','Iron Seed'}

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's main claims: improving KG-RAG effectiveness, efficiency, and transferability through reinforcement learning over structured KG interactions. These claims are supported by experimental results in Section 4.3–4.5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix A discusses dependence on KG quality and coverage, variation in transfer performance across graph structures and relation vocabularies, the assumption of benchmark-provided linked topic entities, and the computational cost of RL training.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Remark 3.1 states the path-realizability result for the KG action set. Appendix B provides the operator definitions and proof, and discusses how the same interface supports execution on different KGs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The main paper and appendix describe the retrieval actions, reward formulation, training procedure, hyperparameters, and evaluation protocol. The public code repository is linked in the abstract.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The public repository linked in the abstract provides the implementation, training scripts, and evaluation pipelines. The benchmark datasets are publicly available and cited in Appendix C.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify the benchmark datasets, model backbones, training horizon, reward settings, RL optimization configurations, and evaluation metrics in Section 4 and the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The main performance tables and ablation studies report means and standard deviations over five evaluation runs. These standard deviations measure run-to-run evaluation variation. The SFT and GNN-RAG comparisons are single-run evaluations, as stated in their table captions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide hardware specifications, memory configurations, precision settings, and training duration details in the appendix, including GPU type and model-specific resource allocation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that this work complies with its guidelines. The experiments use publicly available datasets and models without involving human subjects or sensitive personal data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss both positive societal impacts and potential negative impacts in Appendix A, including improved factual reasoning and risks of propagating incorrect or manipulated knowledge.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not release high-risk models or sensitive datasets. The proposed method builds upon existing open-weight language models and public benchmark datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets, models, and baseline implementations used in this work are properly cited, and their original licenses and usage terms are respected.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release the implementation of KG-R1 with documentation and scripts for reproducing the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This work does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects and does not require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Large language models are a core component of the proposed KG-R1 framework and are fully described in the methodology section. Additionally, we used an LLM-based coding assistant to support parts of the implementation. All generated code was reviewed and validated by the authors, and all experiments were designed, executed, and verified by the authors.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.