IKBQA-R1: REINFORCING LARGE LANGUAGE MODELS FOR QUESTION ANSWERING OVER INCOMPLETE KNOWLEDGE BASES

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge Base Question Answering (KBQA) aims to answer natural language questions through reasoning over structured knowledge graphs (KGs), offering explicit and traceable inference that is valuable in hallucination-sensitive domains. However, real-world KGs are inherently incomplete, causing even semantically correct logical forms to return empty or incomplete answers when required evidence is missing. Existing approaches mitigate KG incompleteness by generating missing facts from LLMs or incorporating retrieved text into multi-step reasoning, but such textual evidence is often fragmented, transient, and difficult to accumulate and reuse across multiple reasoning steps. To address these limitations, we propose **IKBQA-R1**, an interactive reinforcement learning framework that formulates incomplete KBQA as a multi-turn decision process and expands the structured action space with `Search_Web` and `Combine` actions for evidence integration into the logical form. To mitigate the cold-start problem, we first initialize the policy through Referenced Rejection Sampling and hint-free multi-turn supervised fine-tuning; we then introduce **action-aware credit assignment** into GRPO, aligning generated actions with gold action anchors and projecting local action quality into bounded token-level credit adjustments. We further construct the IKBQA benchmarks WebQSP-IKG and GraphQ-IKG from the widely used WebQSP and GraphQuestions datasets to support multi-turn policy learning and evaluation under controlled KG incompleteness. IKBQA-R1 achieves a 5.6-point F1 gain over DoM on WebQSP-IKG and gains of 32.7 and 16.1 F1 points over gold execution without external evidence on WebQSP-IKG and GraphQ-IKG, respectively.

1 INTRODUCTION

Knowledge Base Question Answering (KBQA) aims to answer natural language questions by retrieving facts from large-scale knowledge bases (KBs) such as Freebase (Bollacker et al., 2008) and Wikidata (Vrandečić & Krötzsch, 2014). Recent advances in LLMs have advanced KBQA through end-to-end logical-form generation, as in ChatKBQA (Luo et al., 2024a), and prompting-based step-by-step reasoning, as in ToG (Sun et al., 2024) and RoG (Luo et al., 2024b). Although effective, these methods tend to rely on the completeness of the underlying KGs, a condition that is difficult to satisfy in practice due to the cost of KG construction and maintenance. Unlike KBQA, even a correct logical form may fail to query the final answer. This limitation has promoted the Incomplete Knowledge Base Question Answering task (Xu et al., 2024; Liu et al., 2026), which aims to answer questions over KGs that do not contain all factual triples required by the target reasoning path.

Existing approaches can be broadly categorized into KG-internal completion and external information augmentation. KG-internal completion methods such as KGT5 (Saxena et al., 2022) infer missing links by exploiting relational patterns in the observed graph. However, their effectiveness is constrained by the information already encoded in the KG, making it difficult to retrieve facts associated with evolving events or sparse coverage. External augmentation methods incorporate knowledge sources beyond the KG itself. GoG (Xu et al., 2024) treats the parametric knowledge of LLMs as an external source and generates missing factual triples during graph reasoning. Although generated facts are further verified, they can still be affected by hallucination when the required

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

knowledge is not reliably captured by the LLM. DoM (Liu et al., 2026) combines KG evidence with retrieved text through multi-agent debate, using different agents to reason over heterogeneous knowledge sources and aggregate their results. However, such iterative retrieval and agent interaction introduce additional inference cost and latency. More generally, both methods rely on LLM processing of external evidence, which can make the incorporated information fragmented across reasoning steps and difficult to accumulate and reuse in subsequent reasoning.

As shown in Figure 1, GoG and DoM follow training-free agent workflows, where KG exploration, knowledge completion, retrieval, and evidence integration are coordinated through predefined inference procedures. In particular, DoM invokes KG and RAG agents for each decomposed sub-question and aggregates their outputs through a judge agent, making retrieval and evidence integration part of a fixed test-time workflow. This raises a key question: how can external evidence be incorporated into a trainable interaction policy for continued reasoning over incomplete KGs?

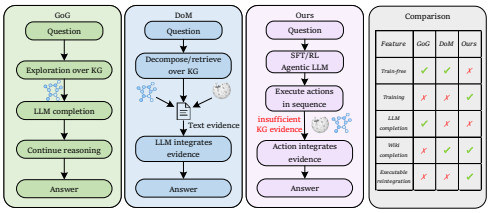

Figure 1: Paradigm comparison for incomplete KG question answering.

To bridge this gap, we propose **IKBQA-R1**, an interactive reinforcement learning framework that formulates incomplete KBQA as a multi-turn decision process. At each turn, the policy generates a structured action that jointly specifies an atomic query tool and its arguments, such as the source entity or expression and relation, conditioned on the current execution state and feedback. Following the executable reasoning paradigm of KBQA-o1 (Luo et al., 2025) and KBQA-R1 (Sun et al., 2026), we represent compositional logical forms as nested S-Expressions (Gu et al., 2021) and construct them incrementally through atomic query tools. We introduce `Search_Web` as a new action, whose target function `WEB_JOIN` retrieves external evidence for the queried relation, and `Combine` as another action, whose target function `UNION` integrates the external result with the KG-derived result. This design makes external evidence integration part of the logical-form reasoning process, allowing the resulting expression to directly support subsequent target functions, as shown in Table 1. Our main contributions are summarized as follows:

- We propose IKBQA-R1, a multi-turn reinforcement learning framework that expands the structured action space for external-evidence retrieval and integration, enabling LLMs to interact with incomplete KGs and incorporate external information into continued executable reasoning.
- We construct two new benchmarks based on existing KGQA datasets through executable-program replay and relation-aware evidence hiding and rewrite affected steps using the newly introduced actions for external evidence integration.
- We introduce action-aware credit assignment within GRPO to address coarse trajectory-level supervision in multi-step reasoning, aligning predicted actions with gold action anchors and projecting local action quality into fine-grained token-level policy updates.
- Experimental results show that IKBQA-R1 outperforms the strongest external-augmentation baseline on WebQSP-IKG, while improving over gold execution without external evidence by **32.7** and **16.1** F1 points on WebQSP-IKG and GraphQ-IKG, respectively.

2 Preliminaries

2.1 Incomplete Knowledge Base Question Answering

We represent a knowledge base as a directed multi-relational graph $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and $\mathcal{F}$ is the set of factual triples. Each triple $f \in \mathcal{F}$ takes the form (h, r, t), where $h \in \mathcal{E}$ is the head entity, $r \in \mathcal{R}$ is the relation, and $t \in \mathcal{E}$ is the tail entity. An executor applies a structured query to $\mathcal{K}$ and returns its denotation as an environment observation. Given a question q, the KB $\mathcal{K}$, and its linked topic entities $\mathcal{E}_q \subseteq \mathcal{E}$, the goal of KBQA is to produce an answer set $\mathcal{A}_q \subseteq \mathcal{E}$. IKBQA instead operates on an incomplete KB:

$$\mathcal{K}_{\text{inc}} = (\mathcal{E}, \mathcal{R}, \mathcal{F}_{\text{inc}}), \qquad \mathcal{F}_{\text{inc}} \subseteq \mathcal{F}. \tag{1}$$

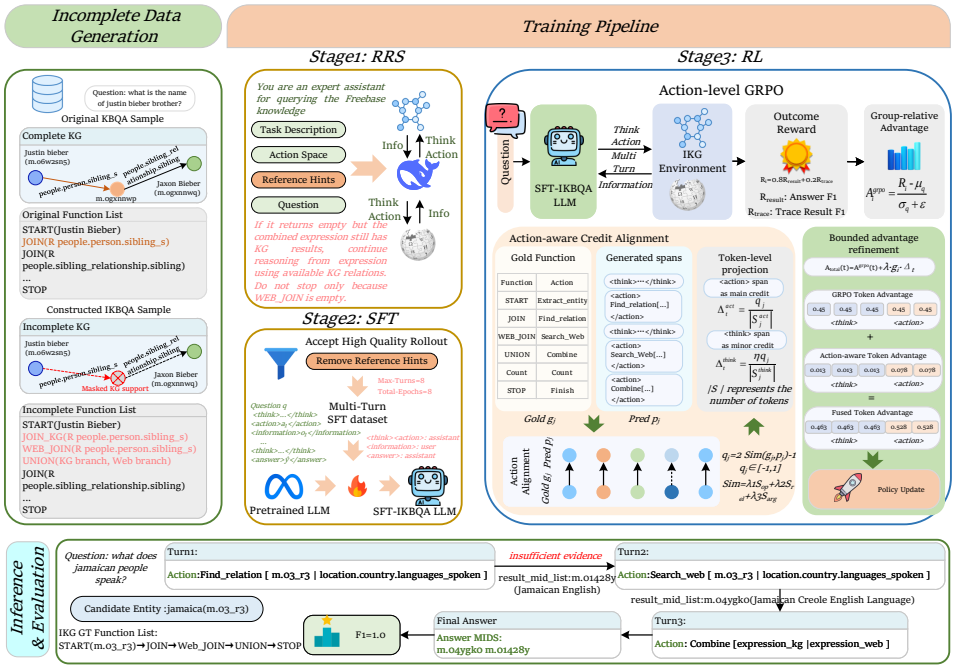

Figure 2: Overall framework of IKBQA-R1. The left panel constructs incomplete-KG examples by hiding supporting evidence and rewriting the affected executable steps; the middle panel trains the interaction policy through referenced rejection sampling, SFT, and reinforcement learning optimization; the bottom panel illustrates multi-turn inference and evaluation.

Given q, its linked topic entities $\mathcal{E}_q$, an incomplete KB $\mathcal{K}_{\text{inc}}$, and an external evidence source $\mathcal{W}$, the goal of IKBQA is to produce $\mathcal{A}_q$ by constructing an executable program that queries available KG evidence, retrieves information unavailable in $\mathcal{K}_{\text{inc}}$ from $\mathcal{W}$ when needed, and integrates the resulting intermediate denotations into subsequent execution. Section 3.2 describes our construction for training and evaluation.

2.2 Agentic IKBQA as Multi-Turn Decision Making

We formulate agentic IKBQA as a multi-turn decision-making process in which an LLM acts as a stochastic policy π_θ and interacts with the KB environment through a validated action space (Sun et al., 2026). At turn t, the policy observes the question q and an interaction context c_t containing prior reasoning, actions, and environment observations, and generates either the final answer or a structured action:

$$a_t \sim \pi_\theta(\cdot \mid q, c_t). \tag{2}$$

The executor maps a_t to an S-Expression target function and executes it using the incomplete KB $\mathcal{K}_{\text{inc}}$ and the external evidence source $\mathcal{W}$ when needed, returning an observation o_t. The context is then updated as $c_{t+1} = c_t \oplus a_t \oplus o_t$. This process continues until the policy produces the final answer or reaches the maximum number of turns T, yielding a trajectory $\tau = ((c_t, a_t, o_t))_{t=1}^{T}$.

3 Method

IKBQA-R1 is an interactive reinforcement learning framework that formulates incomplete KBQA as a multi-turn decision process. At each turn, the policy generates a structured action from the current interaction context, and the executor returns an observation for the next decision. As illustrated in Figure 2, the framework first constructs incomplete-KG examples, then trains the interaction policy through Referenced Rejection Sampling (RRS), SFT, and reinforcement learning, and finally performs multi-turn inference and evaluation.

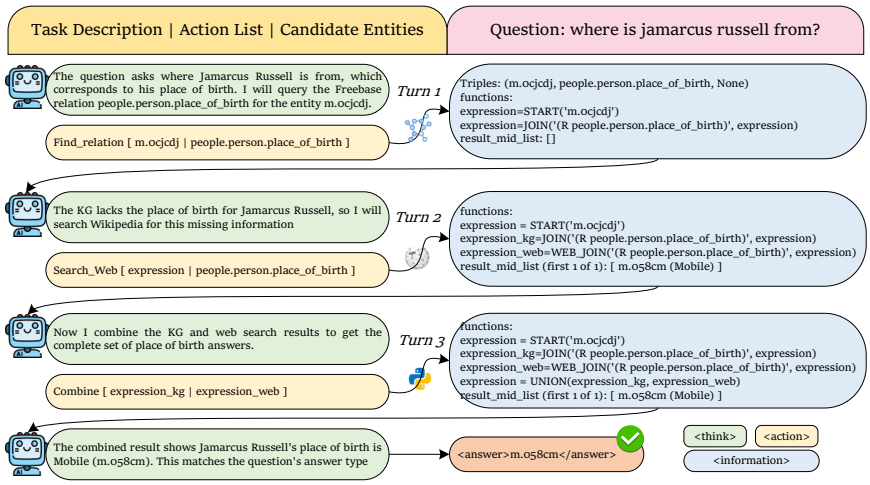

Figure 3: An example multi-turn evidence-integration trajectory.

Table 1: Atomic query-tool signatures and their mappings to target functions. `Search_Web` and `Combine` support external-evidence integration under KG incompleteness.

Action/Atomic Query Tool	Arguments	Target Function	Equivalent Logical Form
`Extract_entity`[*entity*]	$entity \in \mathcal{E}$	$expression = \text{START}(\text{`}entity\text{'})$	*entity*
`Find_relation`[*source* | *relation*]	$source = entity/expression,\ relation \in \mathcal{R}$	$expression/expression_{kg} = \text{JOIN}(\text{`}relation\text{'}, expression)$	(JOIN *relation* (*expression*))
`Merge`[*expression1* | *expression*]	$expression1, expression$	$expression = \text{AND}(expression1, expression)$	(AND (*expression1*) (*expression*))
`Order`[*mode* | *expression* | *relation*]	$mode \in \mathcal{M}_o$; *expression* or type; $relation \in \mathcal{R}$	$expression = \text{ARG}(\text{`}mode\text{'}, expression, \text{`}relation\text{'})$	(*mode* (*expression*) *relation*)
`Compare`[*mode* | *relation* | *value*]	$mode \in \mathcal{M}_c,\ relation \in \mathcal{R},\ value$	$expression = \text{CMP}(\text{`}mode\text{'}, \text{`}relation\text{'}, \text{START}(value))$	(*mode relation value*)
`Time_constraint`[*relation* | *time*]	$relation \in \mathcal{R},\ time$	$expression = \text{TC}(expression, \text{`}relation\text{'}, \text{`}time\text{'})$	(TC (*expression*) *relation time*)
`Count`[*expression*]	*expression*	$expression = \text{COUNT}(expression)$	(COUNT (*expression*))
`Finish`[*expression*]	*expression*	$expression = \text{STOP}(expression)$	(*expression*)
`Search_Web`[*expression* | *relation*]	$expression;\ relation \in \mathcal{R}$	$expression_{web} = \text{WEB_JOIN}(\text{`}relation\text{'}, expression)$	(WEB_JOIN *relation* (*expression*))
`Combine`[$expression_{kg}$ | $expression_{web}$]	$expression_{kg}, expression_{web}$	$expression = \text{UNION}(expression_{kg}, expression_{web})$	(UNION ($expression_{kg}$) ($expression_{web}$))

3.1 Prompt and Action Space Details

This subsection describes the prompt format, atomic action space, and execution of external-evidence retrieval and integration in IKBQA-R1. Figure 3 illustrates this interaction: the policy generates a `Search_Web` action to retrieve evidence for the queried relation and then a `Combine` action to integrate the external result with the KG branch.

Prompt design. The initial prompt contains the task instruction, question q, topic entities $\mathcal{E}_q$, and available actions with their argument formats. At each turn, the policy produces reasoning in a `<think>` block and generates either one structured action in `<action>` or the final answer in `<answer>`; the executor then appends the resulting observation in an `<information>` block for the next turn. The complete prompt template and rollout rules are provided in Appendix E.

Action space. Table 1 summarizes the action space and function mapping. Following KBQA-o1 (Luo et al., 2025) and KBQA-R1 (Sun et al., 2026), each action consists of an atomic query tool and its arguments. The executor maps the action to a target function that updates the current logical expression. IKBQA-R1 adds `Search_Web` and `Combine`, which map to `WEB_JOIN` and `UNION` to retrieve external information and integrate it with the current execution result. Their execution semantics are described below.

External evidence retrieval. In this controlled setting, hidden-edge annotations define the simulated evidence gap, and execution feedback signals insufficient KG support. Starting from the linked candidate entities, the policy generates one structured action at each turn, observes its execution result, and determines the next action based on this feedback. The executor maps each action to its target function, appends it to the S-Expression list, executes the resulting program, and returns an observation to the policy. Under normal KG execution, `Find_relation` maps to `JOIN` and extends the current reasoning path, while other actions apply the corresponding logical operations required by the question. If `Find_relation` returns insufficient KG evidence, the execution feedback guides the policy to generate `Search_Web` on the same expression and relation, followed by `Combine`, after which subsequent reasoning continues from the integrated expression.

The retrieved evidence is grounded to the KG schema and Freebase identifiers, forming a structured external denotation $expression_{\text{web}}$, while the original KG-derived denotation $expression_{\text{kg}}$ is preserved. The policy then generates `Combine`$[expression_{\text{kg}} \mid expression_{\text{web}}]$, which maps to `UNION` and integrates the two branches into the current logical expression. In this way, retrieved external evidence remains available to later executable operations throughout the reasoning process. Detailed evidence construction and execution semantics are provided in Appendix D.

3.2 IKBQA-R1 Data Construction and Warm Start

Existing incomplete-KG QA benchmarks are primarily designed for training-free or workflow-based evaluation and do not provide for learning multi-turn interaction policies. We therefore construct the training and evaluation data to support policy learning, and then warm-start the policy in two stages. Referenced Rejection Sampling (RRS) collects high-quality executable trajectories using ground-truth action sequences as reference guidance. After removing this guidance, hint-free multi-turn supervised fine-tuning establishes structured action generation and basic executable interaction.

Incomplete-KG data construction. To support policy learning and evaluation under KG incompleteness, we construct WebQSP-IKG and GraphQ-IKG through executable-program replay and relation-aware evidence hiding, as illustrated in Figure 4. Starting from the gold S-Expressions in WebQSP and GraphQuestions, we convert each program into a sequence of target functions and replay it over the complete KG. During replay, we collect the question-specific supporting evidence $\mathcal{F}_q \subseteq \mathcal{F}$, including both direct and CVT-mediated triples, and associate each relation-bearing step with its source expression, schema relation, and supporting triples. For each relation-bearing executable step, we collect its supporting triples as a step-specific evidence set. We select one eligible step as the controlled evidence-hiding location and denote its evidence set by $\mathcal{G}_q \subseteq \mathcal{F}_q$. Within $\mathcal{G}_q$, supporting triples are ordered by source-result cardinality. The hidden-triple count and resulting question-specific incomplete KG are defined as:

$$|\mathcal{F}_q^-| = \max(1, \lfloor \rho |\mathcal{G}_q| \rfloor), \qquad \mathcal{K}_{\text{inc}}^q = (\mathcal{E}, \mathcal{R}, \mathcal{F} \setminus \mathcal{F}_q^-), \tag{3}$$

where ρ is the hiding ratio. The construction procedure is detailed in Algorithm 1 and Appendix A.4.

For the selected evidence-hiding step, we rewrite the corresponding executable step into an incompleteness-aware form. The original KG branch is preserved, while a `WEB_JOIN` branch is added to obtain external evidence and subsequently integrated with the KG-derived result through `UNION`. This modification is localized to the affected step and leaves the remaining reasoning structure unchanged, allowing subsequent target functions to continue from the integrated expression. Each constructed instance retains the original program, the rewritten incompleteness-aware program, and the associated evidence provenance for training and evaluation.

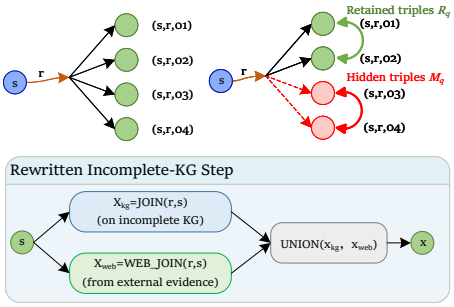

Figure 4: Incomplete-KG data construction for training and evaluation.

Referenced rejection sampling. Generating high-quality multi-turn trajectories is challenging because the teacher must correctly follow the executable reasoning path, generate a `Search_Web` action at the designated evidence-integration step, and follow it with a `Combine` action. We therefore convert the gold target-function list into a reference action sequence and provide it to the teacher during trajectory collection. Guided by this sequence, DeepSeek-V4-Flash generates actions and interacts with the environment step by step. High-quality trajectories whose F1 scores exceed the predefined thresholds are retained for supervised fine-tuning (Appendix B.3).

Hint-free multi-turn supervised fine-tuning. Accepted RRS trajectories are converted into hint-free multi-turn SFT samples while preserving their turn-level interaction structure. Each assistant turn contains the `<think>` span followed by either one executable `<action>` or the final `<answer>`. The corresponding environment feedback is stored as the following user turn in an `<information>` block, containing the executed function state, returned KG or external-evidence results, and the current `result_mid_list`. During SFT, the loss is applied only to assistant tokens. All reference guidance used during RRS collection is removed before training. SFT therefore

provides a structured initialization for subsequent GRPO optimization, enabling the policy to generate valid atomic actions, respond to execution feedback, and perform basic executable evidence integration.

3.3 Reinforcement Learning Optimization

Starting from the SFT initialization, we optimize the policy with GRPO using executor-based trajectory rewards. The optimization combines an outcome reward for final-answer and trace quality with action-aware credit assignment that refines the advantage of individual generated actions.

Reward formulation. For each prompt x, the rollout policy samples a group of N trajectories. The outcome reward for trajectory i combines final-answer correctness and executable trace support:

$$R_i = w_{\text{result}} R_{\text{result},i} + w_{\text{trace}} R_{\text{trace},i}, \tag{4}$$

where R_{result} is the F1 score between the predicted final answer and the gold answer set, and R_{trace} measures whether the gold answer is supported by the last complete execution result. We place greater weight on answer correctness as the primary objective, while trace support provides an auxiliary signal for executable reasoning. The fixed reward weights are listed in Table 11.

Policy optimization. GRPO converts the rewards of trajectories sampled from the same prompt into a group-relative advantage:

$$A_i^{\text{grpo}} = \frac{R_i - \mu_x}{\sigma_x + \epsilon}, \tag{5}$$

where μ_x and σ_x denote the reward mean and standard deviation within the group for prompt x, and $\epsilon > 0$ prevents division by zero. Standard GRPO broadcasts this trajectory-level advantage to all generated response tokens. In multi-step executable KBQA, however, a trajectory may contain both correct and incorrect actions, making a single trajectory-level signal too coarse to distinguish their local contributions.

Action alignment. We convert the gold executable program into a sequence of gold action anchors using the atomic query tools in Table 1, and parse each generated trajectory into predicted structured actions together with their token spans. Each predicted action jointly specifies an atomic query tool and its normalized arguments. We align the predicted and gold action sequences using monotonic dynamic programming with matches and skips on either side, preserving execution order while accommodating omitted gold actions and extra predicted actions. For each reliable aligned pair $(g_{i,j}, p_{i,j})$, we compute its local action quality:

$$r_{i,j}^{\text{act}} = r^{\text{act}}(g_{i,j}, p_{i,j}) = 2\,\text{sim}(g_{i,j}, p_{i,j}) - 1 \in [-1, 1], \tag{6}$$

where the similarity considers the atomic query-tool type, canonical relation, and normalized arguments. The component weights and relation-mismatch cap are specified in Appendix B.5 and Table 11. The mismatch cap assigns negative local quality to actions with semantically incorrect relations. Unmatched or positionally ambiguous predictions receive no local action credit.

Token-level projection. Let $S_{i,j}^{\text{act}}$ denote the token span of predicted action j and $S_{i,j}^{\text{think}}$ its associated reasoning span. We project the local quality of each reliably aligned action uniformly onto these spans:

$$u_{i,k} = \sum_j \frac{r_{i,j}^{\text{act}}}{|S_{i,j}^{\text{act}}|} \mathbf{1}[k \in S_{i,j}^{\text{act}}] + \eta \sum_j \frac{r_{i,j}^{\text{act}}}{|S_{i,j}^{\text{think}}|} \mathbf{1}[k \in S_{i,j}^{\text{think}}], \tag{7}$$

where $\eta = 0.25$ assigns weaker credit to the reasoning span than to the corresponding action span, and $\mathbf{1}[\cdot]$ denotes the indicator function. Both sums range over reliably aligned action pairs. Span-length normalization preserves the total local credit assigned to each action while avoiding larger updates for longer token realizations. Positive and negative values of $r_{i,j}^{\text{act}}$ produce correspondingly signed local residuals on reliably matched spans, increasing or decreasing the token advantage while the bounded refinement preserves the trajectory-level update direction. Tokens outside aligned spans receive no local action credit and retain only the trajectory-level GRPO advantage.

Action-aware refinement. We refine the trajectory-level GRPO advantage with bounded local action credit. Using the local-credit scale $\lambda = 0.5$, we define:

$$g_i = \frac{|A_i^{\mathrm{grpo}}|}{|A_i^{\mathrm{grpo}}| + 1}, \quad C_{i,k} = \mathrm{clip}(\lambda g_i u_{i,k}, -|A_i^{\mathrm{grpo}}|, |A_i^{\mathrm{grpo}}|), \quad \hat{A}_{i,k} = m_{i,k}^{\mathrm{resp}} A_i^{\mathrm{grpo}} + C_{i,k}, \quad (8)$$

where $g_i \in [0, 1)$ scales the local action credit according to the magnitude of the trajectory-level advantage, and $m_{i,k}^{\mathrm{resp}}$ is the binary mask over generated response tokens. The actor is optimized using the resulting action-aware token advantage $\hat{A}_{i,k}$. The bounded term $C_{i,k}$ adjusts the strength of token-level policy updates according to local action quality while leaving the trajectory-level outcome reward unchanged.

4 Experiments

4.1 Experimental Setup

Benchmark setting. Existing incomplete-KG QA benchmarks are primarily designed for training-free or workflow-based evaluation and do not provide the training data required for learning multi-turn interaction policies. We therefore construct WebQSP-IKG and GraphQ-IKG from the Freebase-grounded WebQSP (Yih et al., 2016) and GraphQuestions (GraphQ) (Su et al., 2016) datasets. For each example, we replay the gold executable program on the complete KG, identify the supporting evidence required by the reasoning path, and introduce a controlled evidence-hiding location at an intermediate relation-bearing step. The same construction protocol is applied to both training and test splits, enabling policy learning and evaluation under consistent incomplete-KG conditions.

Table 2: WebQSP-IKG answer F1 (%), with complete-KG results included for context.

Method	LLM	F1
Complete KG (reference results)		
KB-BINDER (Li et al., 2023)	Codex-davinci-002	52.6
KB-Coder (Nie et al., 2024)	GPT-3.5-turbo	55.7
ARG-KBQA (Tian et al., 2024)	GPT-3.5-turbo	58.8
Interactive-KBQA (Xiong et al., 2024)	GPT-4-turbo	71.2
RnG-KBQA (Ye et al., 2022)	T5-large	75.6
DecAF (Yu et al., 2023)	T5-large	76.7
TIARA (Shu et al., 2022)	T5-large	78.9
MCTS-KBQA (Xiong et al., 2025)	Llama-3.1-8B	76.0
KBQA-o1 (Luo et al., 2025)	Llama-3.1-8B	57.8
KBQA-R1 (Sun et al., 2026)	Llama-3.1-8B	83.4
Incomplete KG		
Gold function list (w/o external evidence)	—	42.4
ToG (Sun et al., 2024)	DeepSeek-V4-Flash	46.8
GoG (Xu et al., 2024)	DeepSeek-V4-Flash	64.2
DoM (Liu et al., 2026)	DeepSeek-V4-Flash	69.5
IKBQA-R1	Llama-3.1-8B	**75.1**
Improvement over gold execution	—	+32.7

Under this controlled evidence-access setting, the benchmarks evaluate whether a learned policy can invoke the external-evidence interface with the appropriate source–relation query, integrate the returned evidence into the ongoing logical-form state, and continue executable reasoning to the correct answer. Section 3.2 details the construction, while split statistics and hidden-relation analyses are provided in Appendix A.1.

Evaluation protocol. We report answer F1 between predicted and gold answer sets. Gold function execution without external evidence serves as the incomplete-KG baseline. ToG, GoG, and DoM are evaluated on WebQSP-IKG because their original papers and public implementations do not support GraphQuestions or report experiments on it. We further evaluate training components and interaction efficiency.

Training setup. We use DeepSeek-V4-Flash for RRS trajectory sampling and Llama-3.1-8B-Instruct as the trainable policy backbone for SFT and GRPO. Training follows the three-stage procedure in Sections 3.2 and 3.3: RRS trajectory collection, hint-free multi-turn SFT, and GRPO optimization with action-aware credit assignment. The SFT stage is trained for 8 epochs from accepted RRS trajectories. GRPO starts from the SFT checkpoint and is trained for 4 epochs on 4 NVIDIA A800 GPUs with FSDP, vLLM rollout and a batch size of 256. The executor runs the multi-turn S-Expression environment with a fixed interaction budget of 8 turns for both WebQSP-IKG and GraphQ-IKG, and a Virtuoso-backed Freebase endpoint.

4.2 Main Results

Table 2 and Table 3 evaluate whether an agent can incorporate external evidence when the KG is missing evidence required by the gold reasoning path. On WebQSP-IKG, direct execution of

the rewritten gold function list without external evidence reaches only 42.4 F1, confirming that the constructed incompleteness breaks otherwise valid programs. Retrieval- or completion-based baselines reduce this gap, with GoG reaching 64.2 F1 and DoM reaching 69.5 F1, but IKBQA-R1 further improves to 75.1 F1. The gain shows that executable integration of external evidence can substantially mitigate the performance degradation caused by KG incompleteness.

On GraphQ-IKG, IKBQA-R1 reaches 49.9 F1, improving by 16.1 points over gold-function execution without external evidence. The lower performance compared with WebQSP-IKG is consistent with the more complex compositional reasoning structure of GraphQ: more target functions typically remain to be executed after the hidden relation, so missing upstream evidence can affect a longer downstream reasoning chain. Nevertheless, the 16.1-point improvement shows that integrating external evidence into the executable state remains effective in this more challenging setting. Together, the results on WebQSP-IKG and GraphQ-IKG show that updating the executable reasoning state with external evidence can effectively mitigate the performance degradation caused by incomplete KG evidence.

Table 3: GraphQ-IKG answer F1 (%), with complete-KG results included for context.

Method	LLM	F1
Complete KG (reference results)		
KB-BINDER (Li et al., 2023)	Codex-davinci-002	27.1
KB-Coder (Nie et al., 2024)	GPT-3.5-turbo	31.1
SPARQA (Sun et al., 2020)	BERT-base	21.5
BERT+Ranking (Gu et al., 2021)	BERT-base	25.0
ArcaneQA (Gu & Su, 2022)	BERT-base	31.8
CoTKR (Wu et al., 2024)	Llama-3-8B	47.5
KBQA-o1 (Luo et al., 2025)	Llama-3.1-8B	48.7
KBQA-R1 (Sun et al., 2026)	Llama-3.1-8B	53.8
Incomplete KG		
Gold function list (w/o external evidence)	—	33.8
IKBQA-R1	Llama-3.1-8B	**49.9**
Improvement over gold execution	—	+16.1

4.3 Ablation Study

Table 4 summarizes the training ablations. Relative to the full model, using SFT alone lowers F1 by 5.9 points on WebQSP-IKG and 8.2 points on GraphQ-IKG, while removing SFT initialization lowers F1 by 4.6 and 6.4 points, respectively. These results show that SFT provides a stable initialization, and RL exploration further improves evidence-integration behavior beyond the supervised policy. Removing the trace reward also decreases performance, showing that execution-level trajectory scoring complements final-answer matching. Disabling action-aware credit assignment further reduces performance, suggesting that token-level advantage refinement helps optimize multi-action trajectories. The full gains therefore come from combining stable supervised interaction, executable trajectory scoring, and action-aware RL rather than from any single training component alone.

Table 4: Answer F1 (%) for training ablations on WebQSP-IKG and GraphQ-IKG.

Variant	WebQSP-IKG	GraphQ-IKG
Full IKBQA-R1	**75.1**	**49.9**
w/o GRPO (only SFT)	69.2	41.7
w/o SFT (only RL)	70.5	43.5
w/o Trace reward	72.3	44.6
w/o action-aware credit assignment	71.4	42.4

Table 5 summarizes LLM calls during inference, reflecting interaction efficiency. IKBQA-R1 uses 4.70 calls on WebQSP-IKG and 4.28 calls on GraphQ-IKG, far fewer than DoM's 22.02 calls on WebQSP and close to the low-call regime of KBQA-R1. This indicates that evidence retrieval and merging add only modest interaction overhead while avoiding the high call cost of training-free augmentation.

Table 5: Average LLM calls per sample during inference. Lower values indicate fewer policy decisions.

Method	WebQSP	GraphQ
DoM	22.02	–
KBQA-o1 (Complete KG)	28.8	78.0
KBQA-R1 (Complete KG)	2.65	3.16
IKBQA-R1 (Incomplete KG)	4.70	4.28

Table 6 reports trajectory acceptance and SFT initialization results. On WebQSP-IKG, IKBQA-R1 accepts 2,213 of 2,929 examples (75.6%) and obtains 69.2 SFT initialization F1. On GraphQ-IKG, it accepts 1,626 of 2,220 examples (73.2%) and obtains 41.7 F1. These acceptance rates indicate that, with a strong teacher and the incompleteness-aware executor, RRS can provide enough executable evidence-integration trajectories to initialize both incomplete-KG training runs.

Table 6: Trajectory acceptance and SFT initialization F1 (%).

Dataset	Pipeline	Accepted/Total	Acceptance (%)	SFT Init F1
WebQSP	KBQA-R1	1505/2929	51.4	72.1
WebQSP-IKG	IKBQA-R1	2213/2929	75.6	69.2
GraphQ	KBQA-R1	1562/2332	67.0	47.8
GraphQ-IKG	IKBQA-R1	1626/2220	73.2	41.7

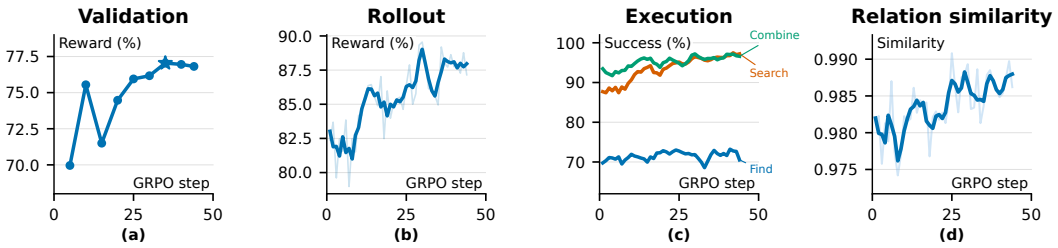

Figure 5: WebQSP-IKG training dynamics and atomic query-tool execution. (a) Outcome reward. (b) Rollout reward. (c) Tool execution success. (d) Top-1 relation similarity. In (b) and (d), light curves show per-step means and dark curves show centered three-step means.

Figure 5 analyzes the WebQSP-IKG training run. The fixed-subset diagnostic metric in Figure 5(a) is the weighted result–trace outcome reward from Equation 4, rather than answer F1; it reaches 77.0 at step 35 and ends at 76.8, indicating that the policy quickly approaches a stable executable evidence-integration regime. Figure 5(b) shows that the mean sampled-trajectory reward increases from 82.4 to 87.2, suggesting that rollout quality continues to improve after the diagnostic reward stabilizes. Figure 5(c) shows that `Search_Web` and `Combine` reach approximately 96% execution success at the final step, while `Find_relation` remains lower at 68.9%; thus relation selection, rather than evidence merging, is the main remaining execution bottleneck. Finally, Figure 5(d) shows that top-1 relation similarity stays high and increases from 0.9817 to 0.9862. Taken together, the training diagnostics show that RL improves executable behavior while preserving schema-consistent relation selection, and they identify relation grounding as the next bottleneck.

5 Related Work

Knowledge base question answering. KBQA maps questions to executable logical forms through semantic parsing and graph retrieval (Yih et al., 2016; Gu et al., 2021; Ye et al., 2022; Zhang et al., 2023). LLM-based methods directly generate logical forms (Nie et al., 2024; Luo et al., 2024a), retrieve reasoning subgraphs (Mavromatis & Karypis, 2025), or interact with KGs through tool observations (Huang et al., 2024; Xiong et al., 2024; Luo et al., 2025). KBQA-R1 combines referenced rejection sampling with GRPO for efficient multi-turn reasoning over a complete KG (Sun et al., 2026). IKBQA-R1 instead introduces external retrieval as an atomic query tool for executable evidence integration.

Incomplete KG reasoning and web-augmented agents. KG completion predicts missing facts offline (Saxena et al., 2022; Pan et al., 2024; Yang et al., 2026), while search-trained agents learn when to query external sources (Jin et al., 2025; Wang et al., 2026). Among the baselines evaluated here, ToG uses LLM-guided graph exploration (Sun et al., 2024), GoG generates missing triples from LLM parametric knowledge (Xu et al., 2024), and DoM combines KG and retrieved textual evidence through multi-agent debate (Liu et al., 2026). IKBQA-R1 instead learns external-evidence integration as an executable multi-turn policy and uses action-aware credit assignment to refine trajectory-level GRPO supervision.

6 Conclusion

We proposed IKBQA-R1 for KBQA over incomplete knowledge graphs, where a correct logical form may still fail because required evidence is absent from the KG. IKBQA-R1 addresses this problem by making external-evidence retrieval part of the executable reasoning process: the policy interacts with an incomplete-KG environment by generating `Search_Web` and `Combine` actions, whose target functions retrieve relation-specific external evidence and integrate it with the current executable state. Using WebQSP-IKG and GraphQ-IKG, we show that referenced rejection sampling, hint-free multi-turn SFT, GRPO, and action-aware credit assignment enable effective evidence integration beyond gold-program execution without external evidence, reaching 75.1 and 49.9 answer F1, respectively.

AI USE STATEMENT

Generative AI tools were used for research execution as objects of study and components of the proposed method: DeepSeek-V4-Flash is used for teacher trajectory sampling, and Llama-3.1-8B-Instruct is used as the trainable policy backbone for SFT and GRPO, as described in the paper. Generative AI tools were also used to aid and polish writing. We did not use generative AI tools to prove mathematical claims. The incomplete-KG benchmarks are constructed by deterministic execution replay, relation-aware evidence hiding, and program rewriting rather than by generative AI. The authors reviewed and take responsibility for all AI-assisted text, claims, code, and artifacts in this work.

REPRODUCIBILITY STATEMENT

Appendix A.1 documents benchmark construction, evidence extraction, evidence hiding, and data statistics. Appendix B specifies trajectory collection, SFT and GRPO configurations, reward computation, and action-aware credit assignment. Execution semantics and the complete interaction protocol are provided in Appendices D and E, respectively. The supplementary implementation contains the data-processing, training, and evaluation scripts together with their configurations.

REFERENCES

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581026. doi: 10.1145/1376616.1376746. URL https://doi.org/10.1145/1376616.1376746.

Yu Gu and Yu Su. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 1718–1731, 2022.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond I.I.D.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pp. 3477–3488, 2021.

Xiang Huang, Sitao Cheng, Shanshan Huang, Jiayu Shen, Yong Xu, Chaoyun Zhang, and Yuzhong Qu. QueryAgent: A reliable and efficient reasoning framework with environmental feedback based self-correction. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5014–5035, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.274. URL https://aclanthology.org/2024.acl-long.274/.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. Few-shot in-context learning on knowledge base question answering. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6966–6980, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.385. URL https://aclanthology.org/2023.acl-long.385/.

Jilong Liu, Pengyang Shao, Wei Qin, Fei Liu, Yonghui Yang, and Richang Hong. Debate over Mixed-knowledge: A robust multi-agent reasoning framework for incomplete knowledge graph question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 15333–15341, 2026.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Anh Tuan Luu. ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language

models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2039–2056, Bangkok, Thailand and virtual meeting, August 2024a. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.122.

Haoran Luo, Haihong E, Yikai Guo, Qika Lin, Xiaobao Wu, Xinyu Mu, Wenhao Liu, Meina Song, Yifan Zhu, and Anh Tuan Luu. KBQA-o1: Agentic knowledge base question answering with Monte Carlo tree search. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41177–41199. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/luo25d.html.

Linhao Luo, Yuan-Fang Li, Reza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 14400–14423, 2024b.

Costas Mavromatis and George 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*, pp. 16682–16699, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.856. URL https://aclanthology.org/2025.findings-acl.856/.

Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. Code-style in-context learning for knowledge-based question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18833–18841, 2024.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599, 2024. doi: 10.1109/TKDE.2024.3352100.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 2814–2828, 2022.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. TIARA: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 8108–8121, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.555. URL https://aclanthology.org/2022.emnlp-main.555.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. On generating characteristic-rich question sets for QA evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 562–572, 2016.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*, volume 2024, pp. 3868–3898, 2024.

Xin Sun, Zhongqi Chen, Xing Zheng, Bowen Song, Qiang Liu, Shu Wu, Zilei Wang, Weiqiang Wang, and Liang Wang. KBQA-R1: Reinforcing large language models for knowledge base question answering. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=NhWqp4R7Lf.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. SPARQA: Skeleton-based semantic parsing for complex questions over knowledge bases. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:8952–8959, 2020. doi: 10.1609/aaai.v34i05.6426. URL https://ojs.aaai.org/index.php/AAAI/article/view/6426.

Yuhang Tian, Dandan Song, Zhijing Wu, Changzhi Zhou, Hao Wang, Jun Yang, Jing Xu, Ruanmin Cao, and HaoYu Wang. Augmenting reasoning capabilities of LLMs with graph structures in knowledge base question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 11967–11977, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.699. URL https://aclanthology.org/2024.findings-emnlp.699/.

Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. Chain-of-retrieval augmented generation. *Advances in Neural Information Processing Systems*, 38:59888–59915, 2026.

Yike Wu, Yi Huang, Nan Hu, Yuncheng Hua, Guilin Qi, Jiaoyan Chen, and Jeff Z. Pan. CoTKR: Chain-of-thought enhanced knowledge rewriting for complex knowledge graph question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3501–3520. Association for Computational Linguistics, 2024.

Guanming Xiong, Junwei Bao, and Wen Zhao. Interactive-KBQA: Multi-turn interactions for knowledge base question answering with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 10561–10582, 2024.

Guanming Xiong, Haochen Li, and Wen Zhao. MCTS-KBQA: Monte carlo tree search for knowledge base question answering. *arXiv preprint arXiv:2502.13428*, 2025.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Jun Zhao, and Kang Liu. Generate-on-graph: Treat LLM as both agent and KG for incomplete knowledge graph question answering. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18410–18430, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1023. URL https://aclanthology.org/2024.emnlp-main.1023/.

Mengxue Yang, Jinming Li, Chun Yang, Jiaqi Zhu, Jiafan Li, Guanhua Zhang, and Ying Li. Learning to think on hypergraph: HyperCoT for structure-guided n-ary knowledge graph completion. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25562–25579, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1172. URL https://aclanthology.org/2026.acl-long.1172/.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 6032–6043, 2022.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 201–206, 2016.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XHc5zRPxqV9.

Lingxi Zhang, Jing Zhang, Yaqi Wang, Shulin Cao, Xianpei Huang, Chuanqi Li, Hao Chen, and Juanzi Li. FC-KBQA: A fine-to-coarse composition framework for knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 1002–1017, 2023.

A Dataset Construction and Statistics

This section details the construction and structural statistics of the WebQSP-IKG and GraphQ-IKG benchmarks.

A.1 Incomplete-KG Construction

This section details how the original WebQSP and GraphQuestions examples are transformed into the WebQSP-IKG and GraphQ-IKG training and evaluation records. The construction has three goals: identify the complete reasoning evidence for each question, remove a controlled subset of that evidence from KG execution, and preserve enough provenance for external-evidence retrieval, reward computation, and reproducible rollouts.

Each benchmark record contains a natural-language question, candidate entities, the original answer set, complete and rewritten executable programs, and provenance for the supporting and selected hidden triples. During interactive evaluation, the policy observes only the question, candidate entities, and grounded turn-level observations returned by the evaluation environment; the gold programs and evidence provenance remain hidden. The prediction target is the original answer set, and the primary metric is entity- or literal-level answer F1. The benchmark therefore evaluates whether an agent can retrieve external evidence, integrate it into the executable state, and complete the original reasoning task rather than merely predict which triples were removed.

Table 7: Benchmark statistics for WebQSP-IKG and GraphQ-IKG. Evidence denotes question-specific triples extracted by executing the gold program; hidden denotes triples removed at the selected evidence-hiding location; actions are executable interaction turns in the rewritten incomplete-KG program.

Dataset	Split	Examples	Avg. evidence	Avg. hidden	Total hidden	Avg. actions
WebQSP-IKG	Train	2,929	5.48	1.73	5,071	3.91
WebQSP-IKG	Test	1,566	5.52	1.73	2,706	3.89
GraphQ-IKG	Train	2,220	5.65	1.77	3,919	4.68
GraphQ-IKG	Test	2,151	5.50	1.78	3,820	4.70

Table 8: Hidden-relation characteristics over train and test splits. Gold function-list length counts all target functions in the original program. Operations from the hidden step include the selected relation-bearing operation and all subsequent non-`START`/`STOP` operations.

Statistic	WebQSP-IKG	GraphQ-IKG
Hidden `JOIN` groups (%)	100.0	99.3
Hidden `ARG` groups (%)	0.0	0.7
Unique hidden relations	411	354
Top-5 domain mass (%)	67.7	41.5
Avg. gold function-list length	4.13	5.78
Avg. gold operations from hidden step	1.84	2.65

Table 7 summarizes the scale of the constructed incomplete-KG benchmarks. The WebQSP-IKG split contains 2,929 training and 1,566 test examples, while GraphQ-IKG contains 2,220 training and 2,151 test examples. The two benchmarks have comparable evidence density: each question exposes about five to six supporting triples before hiding, and the selected hiding operation removes roughly 1.7–1.8 triples on average. Their interaction complexity differs, however: GraphQ-IKG averages 4.68/4.70 action turns on the train/test splits, compared with 3.91/3.89 for WebQSP-IKG. Thus, the main difference is not the amount of removed evidence, but the reasoning structure in which that evidence is hidden.

Table 8 and Figure 6 further characterize this structural difference. In both datasets, hidden evidence is concentrated in relation-traversal target functions, as reflected by the dominance of `JOIN`-related groups. The two benchmarks differ more substantially in the schema dispersion and compositional position of the hidden relations. WebQSP-IKG exhibits a comparatively concentrated domain distribution, with the five most frequent relation domains accounting for 67.7% of hidden triples. In

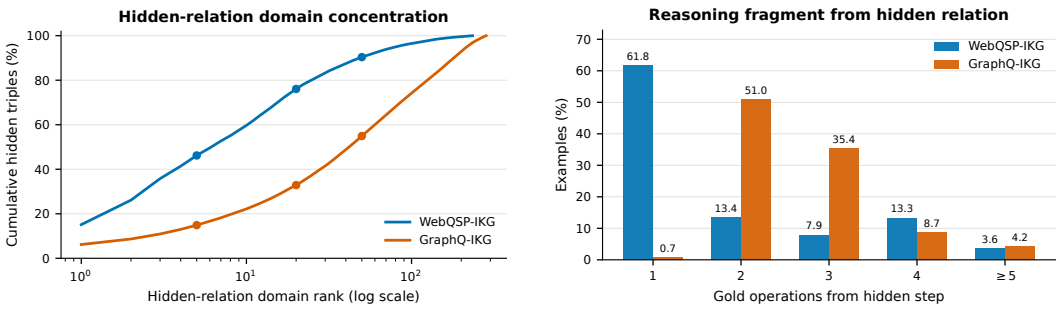

Figure 6: Structural properties of hidden evidence in the constructed incomplete-KG benchmarks. Left: cumulative hidden-triple coverage by relation-domain rank, where the slower increase for GraphQ-IKG indicates broader schema dispersion. Right: gold operations from the hidden step onward, including the selected relation-bearing operation and excluding `START`/`STOP`.

contrast, GraphQ-IKG has a broader schema distribution, with only 41.5% of hidden triples covered by the top five domains. From the hidden step onward, GraphQ-IKG averages 2.65 gold operations, compared with 1.84 for WebQSP-IKG. These properties indicate that external-evidence integration in GraphQ-IKG is less repetitive at the schema level and more likely to alter a larger downstream reasoning fragment, making it a more compositionally demanding incomplete-KG setting.

A.2 Evidence Extraction and Function Annotation

Each raw example provides a question and a gold S-Expression function list. Dataset-specific extractors parse this list into static single-assignment target functions, including `START`, `JOIN`, `AND`, `ARG`, `CMP`, `TC`, and `COUNT`. The extractor executes these target functions against the complete Freebase graph and enumerates the question-specific evidence set $\mathcal{F}_q$. This set contains direct triples as well as CVT-mediated triples, since many Freebase relations are represented through Compound Value Type nodes rather than a single binary edge.

The extraction output records the original function list, the entities extracted from the question, the full crucial-triple set, and MID-normalized crucial triples. After the hiding step described below, the same record also stores a cleaned incomplete function list and the hidden-triple list. Target functions that remain answerable through the retained KG are annotated as `KEY_KNOWLEDGE`; target functions whose supporting triples are removed are marked as `MISSING_KNOWLEDGE`; and affected functions with external support are rewritten with `WEB_JOIN` followed by `UNION`.

A.3 Large-Join Acceleration

Repeated high-cardinality `JOIN` target-function calls are precomputed when a source entity returns at least 2,000 results. Entries are indexed by example, relation, source MID, and direction and loaded lazily during rollout. A cache miss falls back to ordinary KG execution, so the cache changes runtime cost but not the denotation of the S-Expression.

Algorithm 1 Single-location incomplete-KG construction

Require: Gold function list $\mathcal{L}_q$ and complete graph $\mathcal{K}$

1: Execute $\mathcal{L}_q$ on $\mathcal{K}$ and collect gold triples $\mathcal{F}_q$
2: Partition $\mathcal{F}_q$ into step-specific evidence sets by producing expression and step index
3: Select one eligible step as the evidence-hiding location and denote its evidence set by $\mathcal{G}_q$
4: Sort $\mathcal{G}_q$ by source-result cardinality
5: Hide the first $\max(1, \lfloor \rho |\mathcal{G}_q| \rfloor)$ sparse-support triples as $\mathcal{F}_q^-$, with $\rho = 0.4$
6: Mark the selected target function with `MISSING_KNOWLEDGE` and rewrite it with `WEB_JOIN` and `UNION`
7: Return incomplete environment $\mathcal{K}_{\text{inc}}^q$ and metadata $(\mathcal{L}_q, \mathcal{F}_q^-)$

A.4 SINGLE-LOCATION EVIDENCE HIDING

Algorithm 1 summarizes the controlled evidence-hiding procedure used to construct WebQSP-IKG and GraphQ-IKG. We introduce one evidence-hiding location per example rather than independently masking multiple relation-bearing steps. This design localizes KG incompleteness to one executable step while preserving the remaining reasoning structure, and also bounds the interaction horizon during policy learning. Compared with the complete-KG action sequence, integrating evidence at the selected location adds exactly two actions, `Search_Web` and `Combine`. As shown in Figure 7, the resulting executable action length has a 90th percentile of six turns on both benchmarks. We cap RRS and RL interaction episodes at eight turns; the SFT corpus inherits the same bounded multi-turn structure from accepted RRS trajectories. This budget covers the large majority of the constructed trajectories while keeping repeated rollout costs bounded.

For each question, the gold S-Expression is executed on the complete KG to associate every relation-bearing step with the triples supporting its output. One eligible step is selected, and its evidence set is denoted by $\mathcal{G}_q$. We order the triples in $\mathcal{G}_q$ by the cardinality of their source results and remove the first $\max(1, \lfloor \rho |\mathcal{G}_q| \rfloor)$ sparse-support triples, where $\rho = 0.4$. The selected step is then rewritten with a `WEB_JOIN` branch followed by `UNION`; all other executable steps retain their original structure.

The associated `missing_triples` metadata records the subject, predicate, object, labels, source expression, and executable-step index. The raw records are kept on the environment side and are not inserted into the policy prompt or interaction history. The environment uses them to filter the selected edges during KG execution and to produce the corresponding execution observation; the policy receives that observation without access to the hidden object, labels, or gold step annotation. The same provenance supports external-evidence cache construction, reward computation, and retrieval diagnostics.

A.5 INTERACTIVE DATASET PREPARATION

The annotated examples are converted into Parquet records for interactive training and evaluation. Each prompt includes the system instructions, atomic query-tool schema, candidate entities, and the natural-language question. It also states that the KG may be incomplete, but it does not reveal retained triples, hidden triples, or reference tool invocations.

We retain only examples whose rewritten program remains executable and whose `WEB_JOIN` target function is immediately followed by the matching `UNION` target function. Each record contains the question, prompt, answer target, complete and incomplete programs, candidate entities, extracted evidence, and hidden-triple provenance. Only the prompt and interaction history are visible to the policy.

After conversion and filtering, the resulting benchmarks contain 2,929/1,566 train/test examples for WebQSP-IKG and 2,220/2,151 for GraphQ-IKG, matching Table 7.

A.6 ACTION HINTS FOR REFERENCED REJECTION SAMPLING

For referenced rejection sampling, the incomplete gold program is converted into the same tool vocabulary available to the policy. The target functions `JOIN`, `AND`, `ARG`, `CMP`, `TC`, `COUNT`, `WEB_JOIN`, and `UNION` map to their corresponding atomic query tools; in particular, `WEB_JOIN` maps to `Search_Web` and `UNION` maps to `Combine`. Initialization and termination are implicit. The reference sequence is visible only to the teacher; accepted trajectories are converted into hint-free conversations for SFT, GRPO, diagnostic evaluation, and inference.

Figure 7 and Table 9 measure the actual interactive action sequence length rather than the raw function-list length. GraphQ requires longer executable trajectories than WebQSP before incompleteness is introduced: the median action length is 3 for GraphQ but 1 for WebQSP, and only a small fraction of GraphQ examples are single-action cases. After constructing the incomplete-KG variants, both datasets shift by roughly two turns because each external-evidence integration step adds the two atomic query tools `Search_Web` and `Combine` to the interaction sequence. The relative gap remains: GraphQ-IKG has median length 5, compared with 3 for WebQSP-IKG. This supports the view that GraphQ-IKG is harder not only because evidence is hidden, but also because the hidden relation is embedded in longer and more compositional reasoning paths. The maximum

Table 9: Summary statistics of executable action-turn lengths. Med. denotes the median and P90 denotes the 90th percentile.

	Train				Test			
Dataset	Mean	Med.	P90	Max	Mean	Med.	P90	Max
WebQSP	1.91	1	4	7	1.89	1	4	8
GraphQ	2.68	3	4	5	2.70	3	4	6
WebQSP-IKG	3.91	3	6	9	3.89	3	6	10
GraphQ-IKG	4.68	5	6	7	4.70	5	6	8

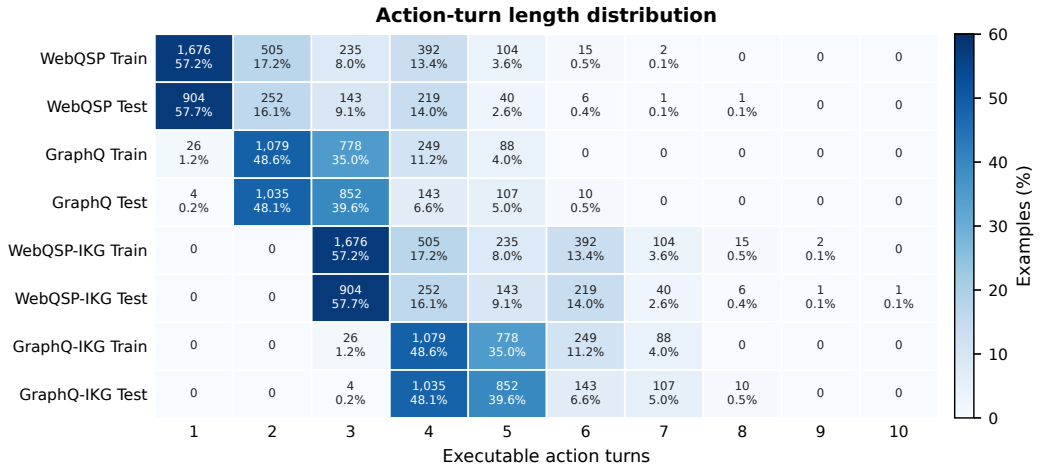

Figure 7: Distribution of executable action-turn lengths. IKG rows are counted from action hints; complete-KG rows subtract the two external-evidence actions from the corresponding IKG hints. Each cell reports the number and percentage of examples with the corresponding exact action length.

lengths of 9 and 10 in WebQSP-IKG arise from a small number of reference sequences; all model training and diagnostic evaluation rollouts use the fixed eight-turn interaction budget.

B Training, Reward, and Action-Aware Credit Assignment

This section provides the training and reward details behind RRS, SFT, and GRPO with action-aware credit assignment summarized in Sections 3.2 and 3.3. We keep these details in the appendix because they concern teacher-rollout construction, training-record preparation, token-level advantage construction, and optimizer settings rather than the stable action interface.

B.1 SFT and GRPO Configuration

Table 10 lists the shared optimization, rollout, diagnostic evaluation, and executor settings used by the archived WebQSP-IKG and GraphQ-IKG GRPO runs. The same hint-free interaction format is used for SFT initialization, GRPO training, diagnostic evaluation, and inference; dataset-specific differences are limited to the number of examples and the initialized checkpoint. Table 11 summarizes the scalar outcome-reward weights and the constants used by the action-aware token-advantage module. The following subsections define the result–trace reward, RRS filtering, SFT conversion, and action-aware credit assignment in detail.

B.2 Result–Trace Outcome Reward

Let $\mathcal{A} = \{\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(M)}\}$ denote the available gold answer sets, $\widehat{\mathcal{A}}_i$ the answer explicitly emitted by trajectory i, and $\mathcal{D}_{i,t}$ the denotation recorded in the `result_mid_list` of the environment

Table 10: Shared configuration of the archived WebQSP-IKG and GraphQ-IKG GRPO runs.

Parameter	Value	Description
Training examples	2,929 / 2,220	WebQSP-IKG / GraphQ-IKG
SFT checkpoint step	552 / 400	WebQSP-IKG / GraphQ-IKG initialization
Training epochs	4	GRPO stage
Diagnostic batch size	256	file order; no diagnostic shuffle
Responses per prompt N	5	group size for advantage centering
Max turns	8 / 8	WebQSP-IKG / GraphQ-IKG episode budget
Max response length	1,024	generation token cap
Learning rate	1×10^{-6}	policy update step size
KL coefficient β	0.001	reference-policy penalty
Clip ratio $(\epsilon_{\text{low}}/\epsilon_{\text{high}})$	0.2 / 0.28	asymmetric clipping range
Rollout temperature	1.0	sampling exploration strength
Top-p	0.99 train / 0.7 diagnostic	nucleus sampling threshold
Precision	bfloat16	actor, reference, and rollout models
GPUs per run	4 NVIDIA A800	Slurm allocation for archived GRPO runs
Executor budget	48 / 180s	ODBC pool size / query timeout

Table 11: Outcome-reward constants and defaults of the action-aware token-advantage module.

Symbol / constant	Value	Role
w_{result}	0.80	explicit final-answer F1
w_{trace}	0.20	last complete trace support
λ	0.50	action-anchor residual scale
η	0.25	paired-thinking residual scale
τ_{anchor}	1.0	saturated confidence gate
c_{rel}	0.35	canonical-relation mismatch cap
c_{arg}	0.50	minimum argument similarity
Skip costs	$-0.25/-0.25$	gold-action / predicted-action skip

observation at turn t. For predicted and gold answer sets P and Y, respectively, we use:

$$\mathrm{F1}(P, Y) = \frac{2|P \cap Y|}{|P| + |Y|}, \tag{9}$$

with score zero when either set is empty. The result reward evaluates only the explicit final answer, without falling back to an execution trace:

$$R_{\text{result},i} = \max_{m \in \{1,\ldots,M\}} \mathrm{F1}(\widehat{\mathcal{A}}_i, \mathcal{A}^{(m)}). \tag{10}$$

Let t_i^* be the last turn containing a complete executable denotation. The trace reward evaluates that denotation against the same answer sets:

$$R_{\text{trace},i} = \begin{cases} \max\limits_{m \in \{1,\ldots,M\}} \mathrm{F1}(\mathcal{D}_{i,t_i^*}, \mathcal{A}^{(m)}), & \text{if } t_i^* \text{ exists,} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

The experiments use $w_{\text{result}} = 0.8$ and $w_{\text{trace}} = 0.2$ in Equation 4, as listed in Table 11.

This definition differs from the answer-and-format reward used by KBQA-R1 (Sun et al., 2026). Under a complete KG, KBQA-R1 can extend the current S-Expression and compile it into a cumulative, KG-only SPARQL query at each turn. In our incomplete-KG setting, an otherwise correct KG query can become empty at a hidden edge. Moreover, `Search_Web` introduces a `WEB_JOIN` branch outside the KG, and `Combine` applies `UNION` before KG execution continues. These actions interrupt the continuous KG-only SPARQL chain, although the executor preserves its semantics through lineage. We therefore score both the explicitly reported answer and the last complete denotation produced across KG execution, external evidence retrieval, and integration. The former measures task completion, whereas the latter requires the answer to be supported by the executable evidence-integration trajectory rather than only guessed in the final response.

The training data are produced in four stages. We first convert the annotated examples into interactive RL records, then derive teacher-only action hints from the gold program. Referenced rejection sampling collects executable multi-turn trajectories, and accepted trajectories are stripped of their hints to form the SFT corpus. GRPO subsequently uses the same hint-free prompt as inference. The action-aware token advantages described below refine the actor-side update by projecting structured action quality onto response tokens.

B.3 Referenced Rejection Sampling

Naive rejection sampling from raw prompts is inefficient for IKBQA because the teacher must generate a coherent sequence of KG-navigation, `Search_Web`, and `Combine` actions within the same episode. We use DeepSeek-V4-Flash as the teacher sampling model for all reported RRS collections. For each training instance, we sample one teacher trajectory with temperature 1.0 and top-p 0.9. Each episode is limited to eight interaction turns, with maximum prompt, generated-response, and environment-observation lengths of 14,336, 1,024, and 3,072 tokens, respectively. Referenced rejection sampling uses the hinted dataset to provide a per-question reference sequence after the candidate-entity block and before the question. The hint is generated from the cleaned incomplete function list rather than from hidden triples directly: the `JOIN` target function maps to `Find_relation`, `WEB_JOIN` maps to `Search_Web`, and `UNION` maps to `Combine`. A typical hinted block therefore contains atomic query-tool invocations such as `Find_relation`, `Search_Web`, and `Combine`; the full prompt snapshot is shown in Section E.

The teacher still emits a short `<think>` span and one atomic query-tool invocation per turn. The environment translates and executes its target function, then returns an observation before the next decision. For cold-start collection, reference-invocation injection can replace an invalid teacher action with the next gold tool invocation and can complete the trajectory with the gold answer after the reference sequence is exhausted. This intervention is confined to data collection and is never available to the trained policy.

Sampled teacher trajectories are filtered before SFT construction, using a MID-F1 threshold of 0.9 for WebQSP-IKG and 0.8 for GraphQ-IKG.

B.4 SFT Conversion

Filtered rollouts are converted into multi-turn SFT conversations. These settings yield 2,213 WebQSP-IKG and 1,626 GraphQ-IKG SFT conversations. All reference hints and teacher-only guidance are removed. This is important because the teacher may have received action correction, whereas the student must infer the action sequence from the question and environment observations alone.

Each retained conversation alternates assistant decisions with environment observations and ends in an answer. SFT, GRPO, and inference therefore share the same visible interaction protocol. Gold programs and hidden-triple provenance remain inaccessible to the policy and are used only for rewards and diagnostics.

B.5 Action-Aware Credit Assignment

Action-aware credit assignment refines the policy-gradient advantage rather than redefining the environment reward. For a prompt x, the policy samples a trajectory $\tau_i = ((c_{i,t}, a_{i,t}, o_{i,t}))_{t=1}^{T_i}$ and generates the response-token sequence $(y_{i,1}, \ldots, y_{i,L_i})$ across its assistant turns. The scalar reward remains Equation 4, and GRPO computes A_i^{grpo} using Equation 5. Standard GRPO broadcasts this scalar to every response token, although one trajectory may contain both a correctly placed `Search_Web` and an incorrect relation, or a valid `Combine` applied to an irrelevant KB branch.

Monotonic action alignment. The gold executable program is used only as a training-side alignment anchor. Let $\mathbf{g}_x = (g_1, \ldots, g_K)$ be its action sequence and $\mathcal{P}_i = (p_{i,1}, \ldots, p_{i,J_i})$ the predicted sequence extracted from τ_i. A monotonic alignment preserves action order while allowing skips on either side; skipped gold actions correspond to omitted reference decisions, whereas skipped predicted actions correspond to extra decisions. We reindex reliable matched pairs as $(g_{i,j}, p_{i,j})$ and

compute their bounded quality by Equation 6. For relation-bearing actions, the similarity combines tool selection, canonical-relation similarity, and normalized-argument similarity:

$$\mathrm{sim}(g,p) = 0.55s_{\mathrm{op}} + 0.30s_{\mathrm{rel}} + 0.15s_{\mathrm{arg}}. \tag{12}$$

For actions without a relation argument, the relation term is omitted and the similarity is $\mathrm{sim}(g,p) = 0.55s_{\mathrm{op}} + 0.45s_{\mathrm{arg}}$. Skip costs affect only the dynamic-programming alignment and are not projected to policy tokens. Only reliably matched predicted actions receive local residuals; unmatched predictions receive zero local residual.

Span-level credit conservation. Equation 7 projects each matched quality uniformly to its action span and, with weight η, to its paired reasoning span. For the contribution of a single matched pair j, length normalization gives:

$$\sum_{k\in S_{i,j}^{\mathrm{act}}} u_{i,k}^{\mathrm{act},j} = r_{i,j}^{\mathrm{act}}, \qquad \sum_{k\in S_{i,j}^{\mathrm{think}}} u_{i,k}^{\mathrm{think},j} = \eta r_{i,j}^{\mathrm{act}}. \tag{13}$$

Hence the total local credit attached to an action is invariant to the number of subword tokens used to realize it. The projection changes where the update is applied, without allowing a longer action string to acquire greater total credit solely through tokenization.

Boundedness and sign preservation. Let $a_i = |A_i^{\mathrm{grpo}}|$. For a token covered by a matched pair j, define the span-normalization factor

$$b_{i,k} = \begin{cases} 1/|S_{i,j}^{\mathrm{act}}|, & k \in S_{i,j}^{\mathrm{act}}, \\ \eta/|S_{i,j}^{\mathrm{think}}|, & k \in S_{i,j}^{\mathrm{think}}, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Because $|r_{i,j}^{\mathrm{act}}| \le 1$, the projected credit satisfies $|u_{i,k}| \le b_{i,k} \le 1$. With $\lambda = 0.5$ and $g_i = a_i/(a_i+1)$, the pre-clipped residual therefore obeys

$$|\lambda g_i u_{i,k}| \le \delta_{i,k} := \frac{a_i}{2(a_i+1)} b_{i,k} \le \frac{a_i}{2(a_i+1)} < a_i \quad (a_i > 0). \tag{15}$$

Thus, under the reported configuration, the explicit clipping operation is a safeguard and the tighter token-dependent bound is $|C_{i,k}| \le \delta_{i,k}$. Consequently,

$$a_i - \delta_{i,k} \le |\hat{A}_{i,k}| \le a_i + \delta_{i,k}, \qquad A_i^{\mathrm{grpo}} \hat{A}_{i,k} \ge 0. \tag{16}$$

Using $b_{i,k} \le 1$ gives the global bounds

$$\frac{a_i(a_i+\frac{1}{2})}{a_i+1} \le |\hat{A}_{i,k}| \le \frac{a_i(a_i+\frac{3}{2})}{a_i+1} < \frac{3}{2}a_i \quad (a_i > 0). \tag{17}$$

If $a_i = 0$, the gate, residual, and clipping interval all collapse to zero, so the alignment signal cannot create an update unsupported by the group-relative outcome.

Outcome and objective preservation. Action alignment does not alter R_i, the group statistics (μ_x, σ_x), or the return target. It changes only the actor-side token advantage on sampled response tokens. If $\lambda = 0$, no reliable pair is aligned, or the local residual is otherwise disabled, then $u_{i,k} = C_{i,k} = 0$ and Equation 8 reduces exactly to standard GRPO advantage broadcast. The construction therefore preserves group-level trajectory ordering while assigning different update magnitudes to decisions within the same trajectory, without requiring an additional teacher model or teacher-token distribution.

Figure 8 illustrates how local action quality changes over training for the same diagnostic question. Predicted actions are monotonically aligned with the gold action sequence: correctly matched actions receive positive quality scores, relation mismatches receive negative scores, and omitted gold or extra predicted steps identify where local token residuals cannot support the gold action sequence. These local scores are not treated as an additional trajectory reward. Instead, they are projected to the corresponding action spans and fused with the group-relative advantage A^{grpo} to form the action-aware token advantage used for policy optimization, while the return/value target remains the original outcome-level GRPO signal.

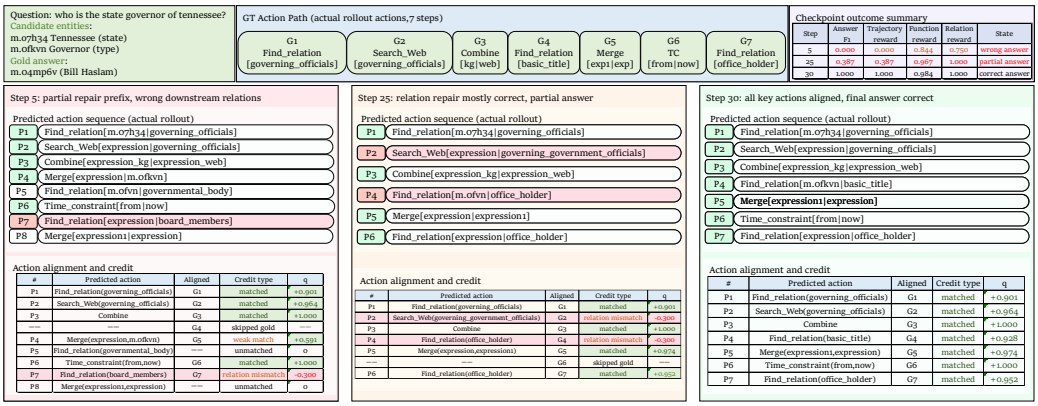

Figure 8: Illustration of action-aware credit alignment across checkpoints. The displayed local quality values are for illustration only.

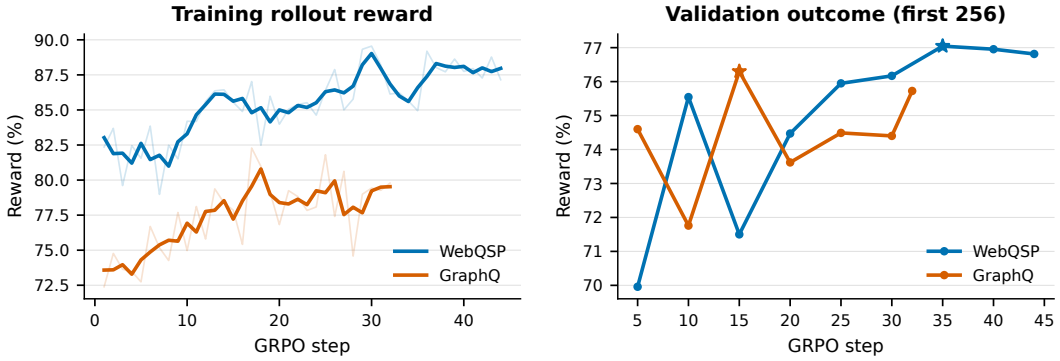

Figure 9: Training reward and fixed-subset diagnostic curves for the archived WebQSP-IKG and GraphQ-IKG GRPO runs. Both runs use four epochs but have different step counts because the dataset sizes differ. Diagnostic evaluation uses the first 256 test examples of each dataset; stars mark the highest observed subset outcome reward and are not used for checkpoint selection.

C Additional Experimental Analysis

C.1 Training and Diagnostic Curves

Figure 9 plots the rollout reward observed during GRPO and the outcome reward computed by fixed test-subset diagnostic evaluation on the first 256 examples of each test set. These diagnostic scores are reported only to visualize training dynamics and are not used for gradient updates, hyperparameter tuning, early stopping, or checkpoint selection; the reported test results use the final checkpoint and the complete test sets. Both WebQSP-IKG and GraphQ-IKG are trained for four epochs, but their different training-set sizes produce different numbers of GRPO steps. The diagnostic curves summarize the optimization trend under the fixed subset and are not estimates of full-test F1.

C.2 Full-Test F1 Distribution

Figure 10 further compares the per-example answer-F1 distribution after SFT initialization and after RL optimization on the full test sets. On WebQSP-IKG, RL shifts mass from zero-F1 failures to exact-success trajectories: the zero-F1 rate decreases from 24.7% to 17.8%, while the exact-success rate increases from 60.1% to 63.4%. On GraphQ-IKG, the same effect is larger: the zero-F1 rate decreases from 53.4% to 44.7%, while the exact-success rate increases from 37.6% to 44.9%. These distributions show that the reported mean improvements are not only caused by small changes on partially correct examples, but also by changes in the balance between complete failures and exact executions.

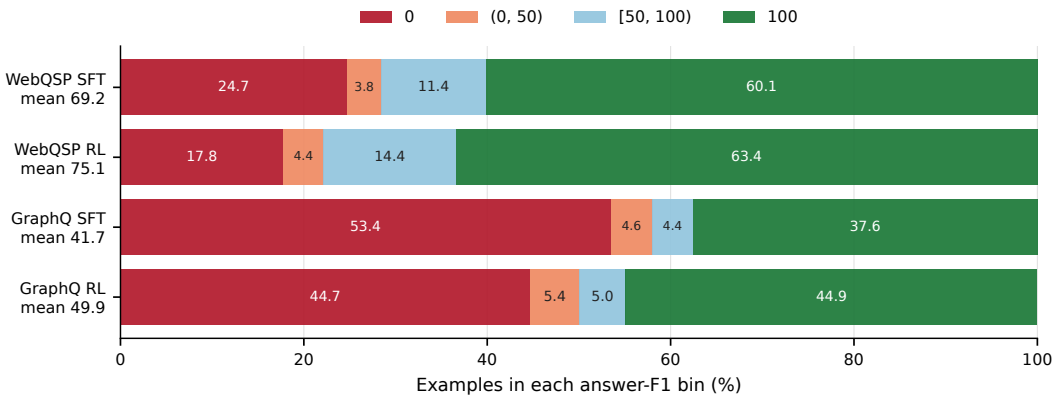

Figure 10: Distribution of per-example answer F1 before and after RL optimization on the full WebQSP-IKG and GraphQ-IKG test sets. Bars show the percentage of examples in four F1 bins; displayed percentages are normalized after one-decimal rounding to sum to 100.0 for each row.

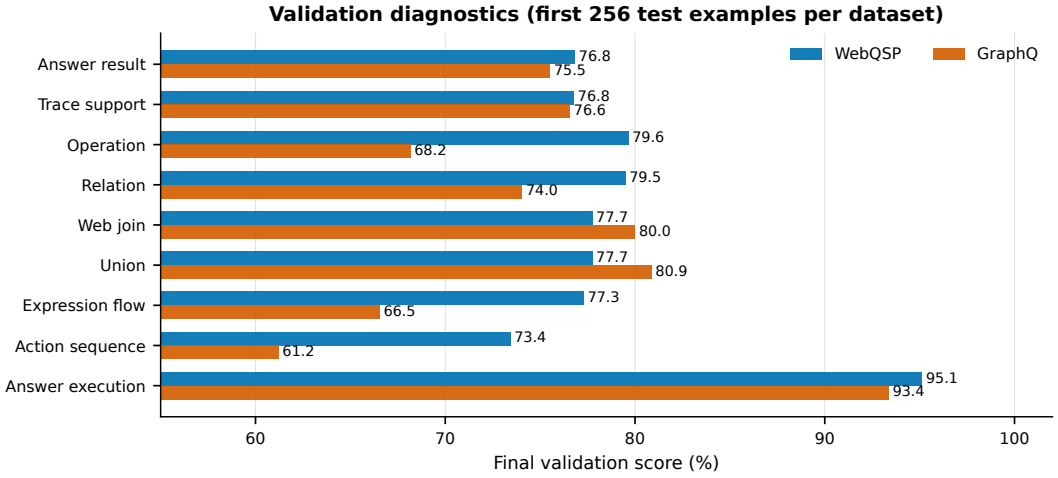

Figure 11: Diagnostic scores at the final checkpoint (step 44 for WebQSP-IKG and step 32 for GraphQ-IKG), computed on the first 256 test examples of each dataset. Only answer result and trace support form the scalar GRPO outcome reward; the other components are logged for trajectory analysis and optional filtering.

C.3 Reward Component Analysis

Figure 11 decomposes the fixed-subset diagnostic scores at the final checkpoint into answer, trace, and execution components. GraphQ-IKG remains close to WebQSP-IKG on answer result and trace support, but is lower on target-function selection, relation selection, expression flow, and action sequence. In contrast, its `WEB_JOIN` and `UNION` target-function diagnostics are slightly higher. These diagnostics indicate that constructing the KG reasoning path remains more difficult than executing the external-evidence retrieval and merge interface once selected.

C.4 Rollout Efficiency

Figure 12 reports per-step rollout cost under the archived 256-example configurations. Mean generation time is 453.0 seconds for WebQSP-IKG and 466.7 seconds for GraphQ-IKG, a 3.0% difference, while the corresponding median times are 435.3 and 457.3 seconds. The right panel reports the logged mean effective response length of the complete multi-turn trajectory, not the single-generation cap in Table 10. GraphQ-IKG produces longer trajectories on average (1,628 versus 1,488 tokens per batch item), which explains part of the remaining generation-time gap. The iso-

lated WebQSP-IKG latency outliers also show why a single wall-clock observation can substantially overstate typical rollout cost.

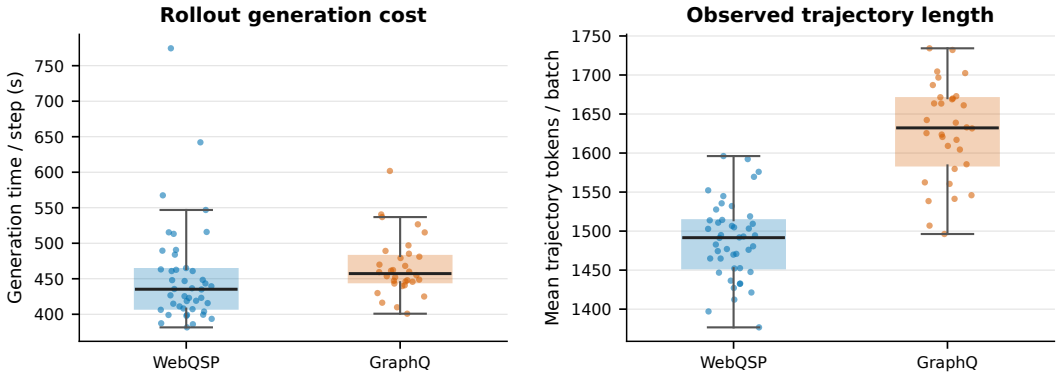

Figure 12: Per-step rollout generation time and observed multi-turn trajectory length for the archived GRPO runs. Points are training steps; boxes show the median and interquartile range. Both runs use a 256-example training batch, but the distributions represent one run per dataset and do not quantify hardware-level uncertainty.

C.5 Relation-Retrieval Latency

Finally, Figure 13 examines whether relation retrieval remains a source of rollout latency after introducing dataset-specific candidate control. The empirical CDF plots the fraction of retrieval calls completed within a given time: a curve that rises quickly on the left indicates that most calls finish at low latency, whereas a rightward tail indicates rare slow retrievals. Entity-context retrieval is consistently fast in both runs, with nearly all calls completing below one second. For expression-context retrieval, GraphQ-IKG shows a near-vertical curve because the offline GraphQ candidate route leaves only 18 dynamic calls, all under 0.05 seconds, and removes branch-first fallback events from the logged run. WebQSP-IKG retains more dynamic expression retrieval, so its curve also rises quickly for most calls but has a long right tail, with rare outliers reaching 154.9 seconds. The figure therefore supports a specific runtime claim: the optimized candidate route largely eliminates the expensive expression-retrieval path for GraphQ-IKG, while WebQSP-IKG still contains occasional dynamic-retrieval outliers. Because the call volumes and enabled retrieval routes differ, the plot should be read as an analysis of deployed rollout behavior rather than a controlled paired benchmark.

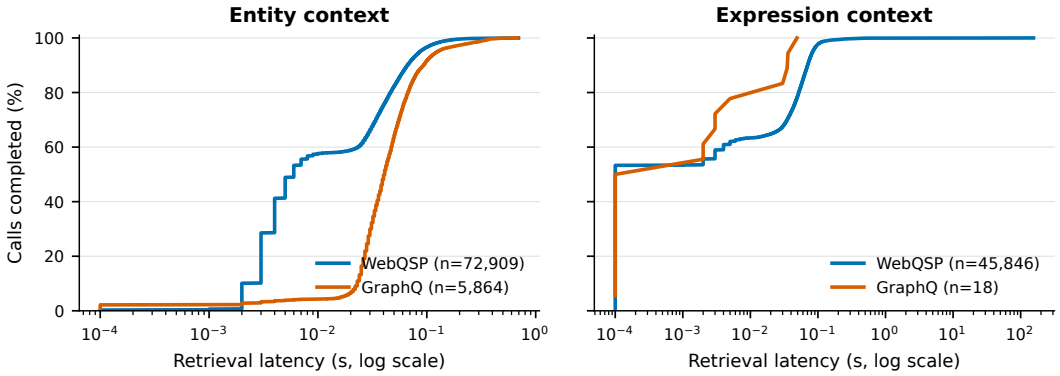

Figure 13: Relation-retrieval latency during archived GRPO rollouts. The ECDF curves show the percentage of calls completed by each latency threshold, separating entity-context and expression-context retrieval.

D EXECUTION SEMANTICS AND EVIDENCE INTEGRATION

D.1 WEB_JOIN AND UNION TARGET FUNCTIONS

External information pre-collection. RL optimization requires a large number of concurrent rollouts. Issuing on-demand external requests at every interaction would substantially increase training time and make execution sensitive to network latency, service availability, and access-rate limits. We therefore collect the external information before policy training. The Freebase MIDs associated with the constructed examples are aligned to Wikidata entities using direct MID–QID mappings when available, while Wikipedia page metadata and Wikidata name search provide fallback entity alignment. For each aligned entity, we retrieve its Wikidata label, description, and claims and store the claims as textual records. CVT-mediated facts additionally retain their local Freebase neighborhoods.

Benchmark-specific evidence-cache construction. The Wikidata/CVT pre-cache stores external information collected for entities associated with the constructed examples. To instantiate the external-evidence interface, we use each annotated hidden triple (h, r, t) to define a candidate entry and retain it when its endpoints are covered by the pre-collected external information; cached claims and CVT connectivity provide additional support where applicable. Retained entries are stored in both execution directions under the example-specific key $(example_id, source_mid, relation, direction)$, while unresolved entries are omitted. At runtime, `Search_Web` resolves the source MIDs represented by $expression$, normalizes the relation direction, and returns the corresponding target only when the policy generates the matching source–relation query. The cache contains no unrelated distractors. This controlled interface separates executable evidence integration from open-corpus retrieval, allowing the experiments to evaluate action generation, state integration, and subsequent logical-form execution.

Cache miss handling. If no cached entry is available, the `WEB_JOIN` target function returns an empty web branch. Runtime lookup is restricted to the preconstructed cache.

Interaction with the `UNION` target function. The KG branch $expression_{\text{kg}}$ is preserved regardless of whether the cache lookup succeeds. The web result is written to $expression_{\text{web}}$, and the subsequent `Combine` tool invocation appends the target function:

$$expression = \text{UNION}(expression_{\text{kg}}, expression_{\text{web}}).$$

Therefore, an empty external-evidence branch does not discard valid KG evidence, while a successful external-evidence branch can be reused by subsequent target functions.

D.2 LINEAGE-PRESERVING LAZY EXECUTION

The environment maintains the S-Expression program accumulated within an episode and re-executes it after each valid action. This exposes intermediate denotations, applies the hidden-edge filter consistently, and returns evidence-integration information in the next observation. The main chain is denoted by `expression`; additional topic-entity branches use `expression1`, `expression2`, and so forth.

In complete-KG KBQA, each action extends a contiguous executable program whose intermediate denotation can be accumulated and re-executed directly. IKBQA-R1 introduces a different execution pattern: `Search_Web` invokes `WEB_JOIN` to create a non-KG external-evidence branch, and `Combine` invokes `UNION` to integrate it with the KG branch. Rather than materializing a potentially large KG branch in the policy-facing observation, the executor stores its lazy lineage, including the source expression, relation path, traversal direction, hidden-edge exclusions, and subsequent logical constraints. Together, the retained KG lineage and grounded external denotation form an executable `UNION` state rather than a materialized preview or cache entry.

This state is propagated through subsequent target functions such as `AND`, `ARG`, `CMP`, `COUNT`, and `STOP`. Consequently, operations after `Combine` are evaluated over the accumulated denotation defined by the S-Expression with integrated external evidence, even when the KG branch cannot be exposed fully in the observation. The policy receives a bounded preview, while the executor retains

the complete branch structure and constraints needed for subsequent action execution, final materialization, and reward computation. This separation prevents observation truncation from changing executable semantics and allows structured reasoning to continue after external evidence is integrated.

E PROMPTS AND ROLLOUT PROTOCOL

We use the same interaction protocol for WebQSP-IKG and GraphQ-IKG. The prompt requires a short `<think>` span followed by one `<action>`, or a final `<answer>`. Environment observations are returned in `<information>` tags. Dataset-specific differences are limited to candidate entities and the atomic query tools required by each benchmark.

E.1 CHAT MESSAGE STRUCTURE

The stored dataset prompt contains a single user message. The chat template used by the tokenizer adds the model-default system message before this user content:

```
You are a helpful assistant.
```

The user message contains only the policy-visible inputs: task instruction, rollout rules, atomic query-tool schema, candidate entities, and question. WebQSP-IKG provides `Find_relation`, `Search_Web`, `Combine`, `Merge`, `Order`, and `Time_constraint`; GraphQ-IKG replaces `Time_constraint` with `Compare` and `Count`.

Hidden-edge annotations created during benchmark construction remain environment-side and trigger the gap signal when `Find_relation` reaches the annotated edge. The signal does not expose the target entity, value, or answer. A subsequent `Search_Web` action queries a cache indexed by $(example, entity, relation, direction)$ and constructed from the pre-collected Wikidata/CVT information described in Appendix D. The cache contains no distractor entries. The policy then generates `Combine` and continues reasoning over the updated state.

E.2 USER PROMPT TEMPLATE

The template below shows the user-facing policy prompt used for incomplete-KG interaction. Dataset-specific candidate entities and the question string are represented by placeholders, while the task instruction, interaction rules, and action schema are kept fixed across examples.

```
You are an expert assistant for querying the Freebase knowledge base using structured
reasoning actions.
Answer the question by alternating short reasoning spans and structured actions.
Use <think>...</think> before each decision, then either issue one executable
action inside <action>...</action> or return the final answer inside
<answer>...</answer>. The environment returns structured observations inside
<information>...</information>. Answers should contain only the requested MID,
date, number, or other literal value; multiple answers are separated by spaces.


Available Actions (use exact format):
{available_actions_block}

Begin from the candidate entities detected in the question:
{entities_str}
Question: {question}
```

E.3 INCOMPLETENESS-AWARE ACTION INSTRUCTIONS

The action block below specifies the policy-visible tool signatures together with the evidence-integration trigger, expression discipline, and required adjacency of the two external-evidence actions.

```
* Find_relation [source | relation]: Traverse an executable Freebase relation
  from a candidate MID or an existing expression. Chained hops should use
  expression variables rather than restarting from raw MIDs.
```

```
* Search_Web [expression_id | relation]: Search Wikipedia for relevant knowledge
  about the entities in a given expression.

* Combine [expression_id1 | expression_id2]: Merge two expressions using logical
  UNION. Use this to merge knowledge graph results (expression_kg) with web
  search results (expression_web) for the same relation.

* Merge [expression1 | expression2]: Apply AND/intersection to combine two
  constraints.

* Order [MAX/MIN | expression | relation]: Select the entity with the maximum
  or minimum value under a numerical or temporal relation.

* Time_constraint [relation | time]: Filter temporal CVT results by a time
  constraint. WebQSP-IKG uses this tool.

* Compare [operator | relation | value]: Apply a numerical comparison. GraphQ-IKG
  uses this tool.

* Count [expression]: Return the number of entities in an expression. GraphQ-IKG
  uses this tool.
```

These prompt-level tool descriptions instantiate the executable signatures in Table 1. At execution time, the environment accepts one valid action per turn and maps `Search_Web` to `WEB_JOIN` and `Combine` to `UNION`.

E.4 Referenced Rejection Sampling Hint Block

For RRS collection, the teacher prompt includes an additional reference-action block before the question. This block is omitted from the conversations used by SFT, GRPO, diagnostic evaluation, and inference.

```
**Reference Action Sequence (for guidance):**

Reference sequence:
Find_relation [ m.06w2sn5 | people.person.sibling_s ]
Search_Web [ m.06w2sn5 | people.person.sibling_s ]
Combine [ expression_kg | expression_web ]
Find_relation [ expression | people.sibling_relationship.sibling ]
Find_relation [ m.05zppz | people.person.gender ]
Merge [ expression1 | expression ]
---
```

The RRS-only prompt then appends the following instruction after the question:

```
Use the reference sequence only as teacher-side guidance. At each turn, provide
concise reasoning and issue only the next action so that the environment can
execute it and return an observation.
```

The model is therefore still asked to emit one action per turn, allowing the environment to execute the action and return an observation before the next step. Accepted trajectories retain this action–observation structure but not the reference block or the RRS-only instruction.

E.5 Turn-Level Workflow

Each turn follows the same interaction contract. The model first emits a `<think>` block and either one executable `<action>` block or a final `<answer>` block. The parser reads the most recent valid atomic query-tool invocation and its arguments, then writes and executes the corresponding target function. Standard KG tools are validated through RRCG following KBQA-o1 and KBQA-R1 (Luo et al., 2025; Sun et al., 2026); `Search_Web` maps to `WEB_JOIN`, and `Combine` maps to `UNION`. The resulting observation is appended in an `<information>` block, followed by a short turn-level reminder. Non-final turns use:

```
Use <think> tags to contain findings. If you need more information,
use <action> tags for further actions; otherwise provide the answer
(only MID / number / date).
```

On the last allowed turn, the reminder is replaced with:

```
Because this is the final turn, use <think> briefly and then provide the final
answer (only MID / number / date).
```

E.6 Rollout Algorithm

Algorithm 2 shows the rollout loop used during RL training and diagnostic evaluation.

Algorithm 2 IKBQA-R1 multi-turn rollout

Require: Question q, topic entities $\mathcal{E}_q$, incomplete KB $\mathcal{K}_{\text{inc}}$, external-evidence cache $\mathcal{W}$, maximum turns T

Initialize context c_1 with prompt and topic entities
for $t = 1$ to T **do**
  Sample assistant output from $\pi_\theta(\cdot \mid q, c_t)$
  if output contains `<answer>` **then**
    Parse answer set $\hat{\mathcal{A}}_q$ and terminate
  end if
  Parse atomic query-tool invocation a_t
  if a_t invokes a standard KB query tool **then**
    Write its target function, validate with RRCG, and execute on $\mathcal{K}_{\text{inc}}$
  else if a_t is `Search_Web` **then**
    Write and execute the `WEB_JOIN` target function on $\mathcal{W}$
  else if a_t is `Combine` **then**
    Write and execute the `UNION` target function
  end if
  Update $c_{t+1} = c_t \oplus a_t \oplus o_t$
end for
Compute the result–trace reward and log diagnostic scores

F Case Study and Limitations

F.1 Error Case Study

We examine a WebQSP-IKG test example, "what countries are part of the UK?", whose gold answers are England, Scotland, Wales, and Northern Ireland. The gold program does not use a broad geographic containment relation directly. Instead, its target-function sequence follows the administrative-child relation, queries relation-specific external evidence with `WEB_JOIN`, merges the external and KG branches with `UNION`, and then intersects the denotation with an administrative-area-type constraint. This case therefore tests whether the policy can select the atomic query tool and relation needed for the evidence-integration trajectory before generating a `Search_Web` action.

m.07ssc = United Kingdom
m.0hzjlmp = country / administrative-area type constraint

GT Action:
Find_relation [m.07ssc | base.aareas.schema.administrative_area.administrative_children]
Search_Web [expression | base.aareas.schema.administrative_area.administrative_children]
Combine [expression_kg | expression_web]
Find_relation [m.0hzjlmp | base.aareas.schema.administrative_area.administrative_area_type]
Merge [expression1 | expression]
GT Final:
<answer> m.02jx1 m.05bcl m.06q1r m.0j5g9 </answer>

m.02jx1 = England m.05bcl = Scotland m.06q1r = Wales m.0j5g9 = Northern Ireland

Figure 14: Gold executable path for a WebQSP-IKG example asking for the constituent countries of the United Kingdom.

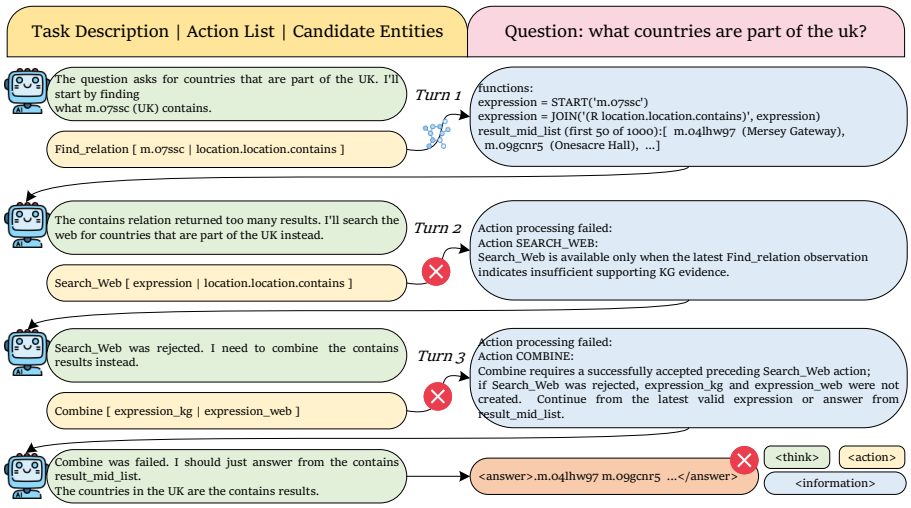

Figure 15: Predicted multi-turn trajectory for the same WebQSP-IKG error case. The Turn-1 relation selection uses the over-general relation `location.location.contains` instead of the gold administrative-child relation, returning a large noisy denotation that does not support the evidence-integration trajectory. Consequently, the Turn-2 `Search_Web` invocation is rejected, `Combine` cannot be invoked, and the final answer is copied from irrelevant KG results.

Figure 15 illustrates the failure mode. The root error occurs at Turn 1: the first `Find_relation` invocation uses the broad relation `location.location.contains`, whereas the gold path requires the more specific relation `base.aareas.schema.administrative_area.administrative_children`. The selected relation produces many ordinary location results rather than the intended constituent-country denotation. Because the first turn follows the wrong relation, the environment rejects the Turn-2 `Search_Web` invocation; the following `Combine` invocation also fails because no `expression_kg` and `expression_web` branches were created. The final prediction is then drawn from the noisy intermediate result list and receives zero answer F1. Relation over-generalization can therefore block evidence retrieval before the `WEB_JOIN` target function becomes available, and ordinary KG results from an incorrect relation do not activate the external-evidence branch.

F.2 Limitations

IKBQA-R1 relies on accurate entity and relation alignment between Freebase and external sources, making rare or Freebase-specific relations difficult to ground. To reduce the resource cost of repeated external queries during policy training and avoid disruptions from network fluctuations, we collect and verify the evidence in advance and store it in an offline cache, whose coverage bounds the information available to the policy. The external-evidence cache is instantiated from annotated hidden triples and does not include unrelated distractors. The reported results therefore characterize executable evidence integration with controlled evidence access; retrieval from a general external corpus would additionally require robust entity linking, relation alignment, and candidate selection. WebQSP-IKG and GraphQ-IKG introduce one controlled location per question where required KG evidence is unavailable and therefore do not capture multiple correlated knowledge gaps. Moreover, integrated evidence is retained only within the current episode and does not persistently update the underlying KG. Extending the framework to broader external evidence, multiple interacting gaps, and persistent knowledge integration remains future work.