# Chain-of-Relations: Faithful and Efficient LLM Reasoning over Knowledge Graphs via Relation-Centric Exploration

**Chenhui Liu, Jianpeng Zhou, Jiahai Wang***
Sun Yat-sen University, Guangzhou, China
{liuchh9, zhoujp7}@mail2.sysu.edu.cn, wangjiah@mail.sysu.edu.cn

## Abstract

Knowledge graph question answering (KGQA) serves as an essential benchmark for KG-enhanced large language models. Among various approaches, agent-based methods have emerged as an effective solution. Existing methods adopt entity-centric exploration that incrementally constructs reasoning paths by selecting and connecting intermediate entities. However, they face two critical limitations. (1) Entity incompleteness vulnerability arises when some intermediate entities lack semantic information beyond opaque IDs, preventing relevance evaluation and leading to discarding valid reasoning paths. (2) Premature entity pruning occurs because beam search retains only top-ranked entities at each step, eliminating candidates before their relevance can be verified. To address these challenges, this paper proposes Chain-of-Relations (CoR)[1] with relation-centric exploration and global entity filtering, reducing dependence on entity completeness and ensuring complete candidate retrieval before constraint validation. Experiments on three benchmark datasets show that CoR consistently outperforms strong baselines in both F1 score and KG-grounded Rate.

## 1 Introduction

Large Language Models (LLMs) have demonstrated remarkable reasoning capabilities across various complex tasks (Zhao et al., 2024). However, three fundamental limitations remain: outdated knowledge (Gao et al., 2023), hallucinations (Huang et al., 2025), and lack of explainability (Zhao et al., 2024).

Knowledge Graphs (KGs) address these limitations through structured, verifiable knowledge representations. This complementary relationship between LLMs and KGs presents opportunities and challenges for faithful AI systems (Pan et al., 2024).

Knowledge graph question answering (KGQA) (Lan et al., 2022) has become a key benchmark for evaluating KG-enhanced LLMs. Previous approaches integrate KGs through pretraining (Zhang et al., 2019; Wang et al., 2021), fine-tuning (Luo et al., 2024), or subgraph retrieval (Baek et al., 2023; Axelsson and Skantze, 2023; Wang et al., 2024). They either rely on static knowledge injection or exclude LLMs from actively participating in the exploration process.

Recently, agent-based approaches (Jiang et al., 2023; Sun et al., 2024; Chen et al., 2024) allow LLMs to actively explore KGs by constructing reasoning paths through entity traversal based on entity semantics. This paradigm is referred to as **entity-centric exploration**.

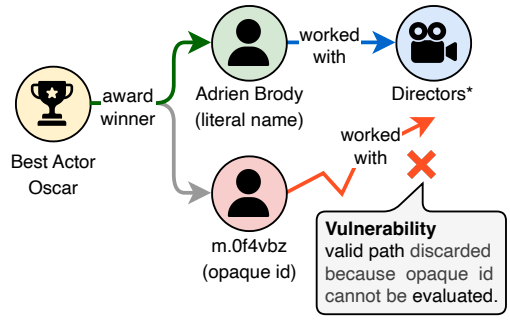
Despite their potential, existing agent-based methods frequently suffer from *exploration failures*, where they fail to retrieve answers from the KG even when target entities are present. As shown in Figure 1, these failures stem from two main limitations:

**(1) Entity incompleteness vulnerability**: Entity-centric exploration builds paths by connecting entities while treating relations as transitional links. This strategy is fragile when intermediate entities lack semantic information, such as those identified only by opaque IDs (e.g., m.0f4vbz) without accompanying literals (e.g., "Adrien Brody"). As shown in Figure 1(a), when answering "Which directors have worked with actors who won the Best Actor Oscar?", an entity-centric approach cannot evaluate the relevance of m.0f4vbz and may skip it entirely. This causes it to miss directors who collaborated with this Oscar-winning actor. **(2) Premature entity pruning**: Entity-centric exploration must aggressively prune entities at each step to maintain computational tractability, typically retaining only

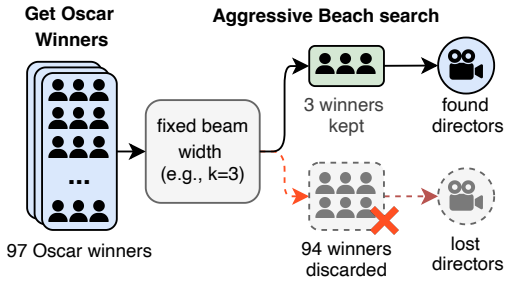
*Corresponding author
[1]Open-source code: https://github.com/Chenhui7au/Chain-of-Relations



*Findings of the Association for Computational Linguistics: ACL 2026*, pages 43108–43119




**(a) Entity incompleteness vulnerability.**
*Path breaks due to lack of semantic information.*




**(b) Premature entity pruning.**
*Loss of answers due to aggressive beam search.*

Figure 1: Two limitations of entity-centric exploration on WebQSP for the question "Which directors have worked with actors who won the Best Actor Oscar?" (a) Entity incompleteness vulnerability: Opaque-ID entities (e.g., m.0f4vbz) cannot be evaluated, causing valid paths to be discarded. (b) Premature entity pruning: A fixed beam width prunes 94 of 97 Oscar winners, excluding directors connected to the pruned actors.

a small subset (e.g., top-$k$ via beam search) for further expansion. The ranking criteria at early stages often rely on surface-level features such as popularity or recency, which do not account for constraints applied in later steps. As illustrated in Figure 1(b), when 97 Best Actor Oscar winners are pruned to only 3 due to a beam width constraint of k=3, 94 actors are discarded. Consequently, directors who exclusively worked with these pruned actors are completely missing from the results. This results in significant recall reduction.

To address these limitations, this paper proposes Chain-of-Relations (CoR), a KG-enhanced LLM reasoning framework with two core components:

(1) **Relation-Centric Exploration** treats relations as the fundamental units for multi-hop reasoning. Instead of connecting entities step-by-step, this paradigm constructs reasoning paths by directly chaining relations (e.g., $\text{relation}_1 \rightarrow \text{relation}_2 \rightarrow \text{relation}_3$). This significantly reduces dependence on entity completeness. By operating in relation space (typically $10^3$–$10^4$ relations) rather than entity space (millions of entities), CoR maintains robust exploration even when intermediate entities lack semantic information. To further enhance fault tolerance, CoR introduces a relation-level backtracking mechanism that automatically reverts to alternative relation paths when encountering dead ends, rather than terminating the exploration. (2) **Global Entity Filtering** postpones constraint filtering until the complete relation chain has been constructed. This decouples relation-chain exploration from entity validation. Unlike entity-centric approaches that incrementally prune entities at each step, CoR first retrieves all reachable candidate entities through the relation chain. It then applies all question constraints simultaneously to validate and select final answers. This global filtering strategy avoids premature elimination of relevant entities.

Additionally, to distinguish between answers derived from structured KG reasoning and those from parametric memory, CoR introduces **KG-grounded Rate**, a metric that quantifies the proportion of answers obtained through explicit KG reasoning paths. This metric enables fine-grained assessment of reasoning faithfulness beyond mere answer correctness, and validates whether improvements arise from genuine KG utilization rather than LLM memorization.

The contributions of this paper are as follows:

- **CoR** is proposed to address entity incompleteness vulnerability and premature entity pruning through Relation-Centric Exploration and Global Entity Filtering.
- **KG-grounded Rate** is introduced to quantify the proportion of answers derived through explicit KG reasoning rather than parametric memory, enabling a fine-grained assessment of reasoning faithfulness beyond mere correctness.
- Extensive experiments on WebQSP, CWQ, and QALD10 demonstrate that CoR achieves significant gains in both correctness (F1 score) and faithfulness (KG-grounded Rate), while requiring fewer API calls.

## 2 Related Work

### 2.1 LLM Reasoning

LLM reasoning methods rely entirely on parametric memory acquired during pre-training. Chain-

of-Thought (CoT) (Wei et al., 2022) prompts LLMs to generate step-by-step reasoning, improving complex problem-solving capabilities. Tree-of-Thought (ToT) (Yao et al., 2023) extends this by exploring multiple reasoning branches in a tree structure, while Graph-of-Thought (GoT) (Besta et al., 2024) organizes thoughts as graphs to capture non-linear reasoning dependencies. However, without access to dynamically updated KGs, these approaches cannot ensure faithfulness to factual knowledge.

### 2.2 Retrieval-based Reasoning

Retrieval-based reasoning methods retrieve relevant knowledge from KGs as context to guide LLM reasoning. KD-CoT (Wang et al., 2023) retrieves factual triples from external KGs to improve the accuracy of chain-of-thought reasoning. RoG (Luo et al., 2024) advances this idea by proposing a planning-retrieval-reasoning framework in which structured reasoning paths are retrieved to improve the faithfulness of LLM reasoning. To better capture graph topology, GNN-RAG (Mavromatis and Karypis, 2024) employs a lightweight neural network for more effective KG retrieval. However, these methods treat KG retrieval as a preprocessing step, excluding LLMs from actively participating in the exploration process and limiting their ability to handle complex multi-hop reasoning.

### 2.3 Agent-based Reasoning

Agent-based reasoning methods employ LLMs as autonomous agents (Jiang et al., 2024; Liu et al., 2025), granting them greater flexibility for active KG exploration. StructGPT (Jiang et al., 2023) designs specialized interfaces to iteratively reason over structured data. ToG (Sun et al., 2024; Ma et al., 2025) incorporates beam search to explore multiple reasoning paths simultaneously. PoG (Chen et al., 2024) proposes a self-correcting adaptive planning framework that decomposes questions into sub-objectives. However, these methods adopt an *entity-centric exploration* strategy that constructs reasoning paths by connecting intermediate entities, making them vulnerable to entity incompleteness during exploration.

## 3 Preliminary

This section formally establishes the foundational concepts and problem formulation for knowledge graph question answering.

**Knowledge Graph.** A knowledge graph is represented as $G = (E, R, T)$. Here, $E$ denotes the set of entities, $R$ denotes the set of relations, and $T$ denotes the set of triples. Each triple $(e_h, r, e_t) \in T$ represents a fact. In this triple, $e_h$ and $e_t$ belong to $E$, and $r$ belongs to $R$.

**Knowledge Graph Question Answering.** Given a question $q$ and a KG $G$, KGQA aims to retrieve answer entities $E_{target} \subseteq E$ from $G$. The process begins by identifying topic entities $E_{topic} \subseteq E$ mentioned in $q$. Starting from $E_{topic}$, the system explores the KG to discover $E_{target}$ that answer $q$. The exploration continues until the system finds valid answers or reaches termination conditions.

**Relation Chain.** A relation chain $c = \{r_1, r_2, \ldots, r_l\}$ is a sequence of relations that defines a reasoning path. Unlike entity-centric paths that alternate between entities and relations (e.g., $e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} e_3$), relation chains consist solely of relations (e.g., $r_1 \rightarrow r_2 \rightarrow r_3$), treating intermediate entities as implicit variables. When executed from topic entities $E_{topic}$, a relation chain retrieves all entities reachable by sequentially following the relations in $c$.

**Question Constraints.** A set of constraints $\mathcal{C} = \{\kappa_1, \kappa_2, \ldots, \kappa_m\}$ specifies requirements that candidate answers must satisfy. Each constraint $\kappa_i$ defines a validation condition that determines whether an entity meets a specific requirement derived from the question. The final answers are obtained by selecting entities from the candidate set $E_{cand}$ that satisfy all constraints in $\mathcal{C}$ simultaneously: $E_{final} = \{e \in E_{cand} \mid e \text{ satisfies all } \kappa_i \in \mathcal{C}\}$. Common constraint types include attribute constraints (e.g., "nationality=American"), temporal constraints (e.g., "year $>$ 2010"), and comparative constraints (e.g., "population $>$ 1M").

## 4 Chain-of-Relations Framework

CoR addresses entity incompleteness and premature entity pruning through two core components: *relation-centric exploration* and *Global Entity Filtering*. Given a question $q$, a knowledge graph $G = (E, R, T)$, and topic entities $E_{topic}$ identified from $q$, CoR finds answer entities $E_{target}$ by constructing a relation chain $c = \{r_1, r_2, \ldots, r_l\}$ that connects $E_{topic}$ to $E_{target}$.

### 4.1 Relation-centric Exploration

At the start of the exploration process, the agent initializes its state as $s^0 = (c^0, h^0)$. Here, $c^0$ is

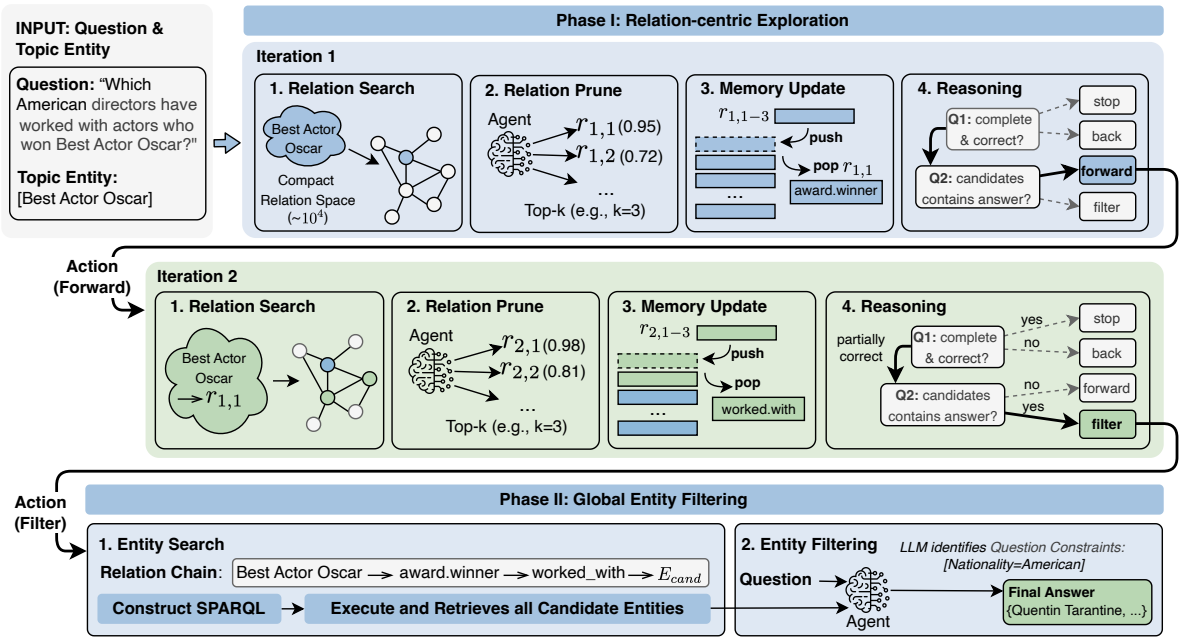



Figure 2: Overview of the Chain-of-Relations framework on an example question from WebQSP: "Which directors have worked with actors who won the Best Actor Oscar?" Phase I constructs reasoning paths by chaining relations from topic entities. Phase II validates candidate answers after complete path exploration.

an empty relation chain and $h^0$ is an empty exploration memory stack. The agent iteratively updates its state at each step $t$ to $s^t = (c^t, h^t)$. Here, $c^t$ is the current relation chain and $h^t$ records the history of explored chains and alternative branches for backtracking. At each iteration, the Relation-centric Exploration operates through four key steps: relation search, relation prune, memory update, and reasoning.

**Step 1: Relation Search** identifies candidate relations that can extend the current reasoning chain without requiring explicit instantiation of intermediate entities. Given the current relation chain $c^t = \{r_1, \ldots, r_t\}$ starting from topic entities $E_{topic}$, the system queries the KG to find all relations $R_{cand}$ that can connect to new entities. Analogous to variable binding in SPARQL queries, entities serve as abstract variables. The system operates directly in the compact relation space (typically $10^3$–$10^4$ relations versus millions of entities). This reduces computational cost while maintaining reasoning flexibility.

**Step 2: Relation Prune** leverages the LLM to score and filter candidate relations based on their relevance to the question and logical consistency with the current chain. For each relation $r \in R_{cand}$, the LLM assigns a relevance score $score(r|q, c^t)$ based on three criteria: (1) *semantic alignment* evaluates whether the relation semantically fits the question intent; (2) *logical consistency* assesses whether the relation is a natural continuation of $c^t$; (3) *goal proximity* estimates whether the relation likely leads toward answer entities. The system selects the top-$k$ relations with highest scores: $R_{prune} = \text{Top-}k(R_{cand}, score)$.

**Step 3: Memory Update** maintains an exploration history that enables backtracking when the current path fails to yield valid answers. After obtaining the top-$k$ pruned relations $R_{prune}$, the system forms candidate relation chains by combining each relation $r_i \in R_{prune}$ with the current chain $c^t$, yielding $\{c^t \cup \{r_i\} \mid r_i \in R_{prune}\}$. These candidates are pushed onto the memory stack $h^t$ in descending order of their relevance scores. This ensures that the most promising chains are explored first via depth-first search while preserving alternative branches for backtracking when needed.

**Step 4: Reasoning** At each iteration $t$, the LLM agent decides the next action based on the current state $s^t = (c^t, h^t)$ and question $q$. The agent can choose from four actions. (1) **Stop** terminates exploration when the relation chain $c^t$ is deemed complete and correct. (2) **Forward** continues exploration by popping the next candidate relation chain from the memory stack $h^t$ when the current chain is partially correct but requires further extension. (3) **Backtrack** reverts to an alternative branch from $h^t$ and continues exploring when the current relation

chain is incorrect or leads to semantic inconsistency with $q$. (4) **Filter** transitions to *Global Entity Filtering* when the current candidate entities are likely to contain the correct answer.

The action planning follows a hierarchical decision-making process. This process is implemented by prompting the LLM with two sequential questions:

**Q1: Is the current relation chain complete and correct?** The agent evaluates the logical alignment of $c^t$ with the question. Three possible outcomes exist. (1) *Correct* triggers **Stop**. (2) *Incorrect* triggers **Backtrack**. In this case, the agent reverts to an alternative relation chain from $h^t$ and continues exploration. (3) *Partially correct* proceeds to Q2.

**Q2: Do the current entity candidates likely contain the correct answer?** The agent evaluates whether the current candidate entities include the correct answer. Two possible outcomes exist. *No* triggers **Forward**, and the system pops the next candidate relation chain from $h^t$ to continue exploration. *Yes* triggers **Filter**, and the system transitions to *Global Entity Filtering*.

This hierarchical decomposition reduces planning difficulty. It also improves action accuracy. By maintaining exploration history in $h^t$, the backtracking mechanism effectively prevents irreversible errors, which enables systematic recovery from early-stage reasoning failures.

### 4.2 Global Entity Filtering

After relation-centric exploration constructs a complete chain $c^{final} = \{r_1, \ldots, r_l\}$, the Global Entity Filtering stage validates candidate entities against all question constraints simultaneously using LLM reasoning.

**Step 1: Entity Retrieval.** The relation chain is executed on the KG to retrieve candidate entities $E_{cand}$. This query returns all entities reachable from $E_{topic}$ via the relation path $c^{final}$.

**Step 2: Entity Filtering.** Given the question $q$ and candidate entities $E_{cand}$, the LLM leverages its reasoning capability to identify constraints within the question and filter $E_{cand}$ to find the final answers. For example, for the question "Which American directors have worked with actors who won Best Actor Oscar?", after executing the relation chain *award.winner* $\rightarrow$ *worked_with*, the LLM identifies the nationality constraint (American) from the question. The LLM then selects only directors from $E_{cand}$ whose nationality is American. This produces the final answer set:

$$E_{final} = \{e \in E_{cand} \mid e \text{ satisfies } \mathcal{C}\}$$

Unlike entity-centric approaches that incrementally prune entities at each hop, global filtering defers constraint validation until completion of path exploration. This prevents premature elimination of candidates that may only satisfy constraints upon reaching later reasoning stages, thereby improving answer recall.

### 4.3 KG-grounded Rate

To assess the faithfulness of KG-enhanced reasoning, *KG-grounded Rate* is proposed as a complementary evaluation metric. Traditional metrics such as Hits@1 and F1-score mainly focus on answer correctness, but they do not distinguish whether answers are derived through explicit KG reasoning or generated from the LLM's parametric memory. KGR addresses this gap by focusing on answer faithfulness.

KG-grounded Rate quantifies the proportion of questions for which the system successfully constructs a valid reasoning path in the knowledge graph. Specifically, let $N_{KG}$ denote the number of questions answered via structured KG traversal. Let $N_{total}$ denote the total number of questions. The metric is defined as:

$$\text{KG-grounded Rate} = \frac{N_{KG}}{N_{total}}$$

An answer is considered KG-grounded if the system successfully constructs a complete relation chain $c^{final}$ and retrieves candidate entities $E_{cand}$ from the KG. The final answer must be derived from $E_{cand}$ through the **Stop** or **Filter** action. This criterion distinguishes answers grounded in external knowledge from those relying on the LLM's parametric memory due to exploration failures. KG-grounded Rate provides a fine-grained assessment of how effectively systems leverage knowledge graphs.

## 5 Experimental Setup

### 5.1 Datasets

This work evaluates CoR on three KGQA datasets from two knowledge graphs: WebQSP and CWQ on Freebase[2], and QALD10-EN on Wikidata[3].

[2]https://developers.google.com/freebase

[3]https://www.wikidata.org/

**WebQSP** (Yih et al., 2016) contains 1,639 test questions, requiring 1–2-hop reasoning on average. Questions involve relatively straightforward reasoning paths with simple constraints.

**ComplexWebQuestions (CWQ)** (Talmor and Berant, 2018) contains 3,531 test questions featuring more complex multi-hop reasoning, with 2–4 hops on average and multiple constraints.

**QALD10-EN** (Perevalov et al., 2022) contains 333 test questions over Wikidata and emphasizes compositional reasoning with diverse constraint types.

### 5.2 Baselines

To provide a comprehensive evaluation of CoR, this paper compares it against a diverse set of strong baselines.

**LLM-only Methods.** IO Prompting performs direct question answering without intermediate reasoning steps. Chain-of-Thought (CoT) (Wei et al., 2022) prompts the LLM to generate step-by-step reasoning with the instruction "Let's think step by step."

**Agent-based Methods.** ToG (Sun et al., 2024) employs entity-centric exploration with beam search. PoG (Chen et al., 2024) uses an adaptive planning framework with a self-correction mechanism. Like CoR, both ToG and PoG are training-free methods. This work implements both ToG and PoG using their open-source code for fair comparison.

### 5.3 Implementation Details

**LLM Configuration.** All methods are evaluated with two backbone models: gpt-4.1-mini and DeepSeek-V3.1. The primary motivation for this selection is to verify that CoR remains effective across different LLM backbones. Specifically, gpt-4.1-mini is a closed-source model with fast inference speed, while DeepSeek-V3.1 is an open-source model optimized for reasoning capabilities but with slower inference speed. Both models use a temperature of 0.01 for deterministic reasoning.

**Agent-based Methods Configuration.** For all agent-based methods including CoR, ToG, and PoG, the exploration process uses top-$k$ pruning with $k = 3$ as the beam width for selecting relations or entities at each step. The maximum reasoning depth is set to $l = 3$ hops on WebQSP and QALD10, and $l = 4$ hops on CWQ.

**Knowledge Graph Setup.** This work uses the official Freebase snapshot following the setup in ToG (Sun et al., 2024). For Wikidata, this work uses the official SPARQL endpoint[4]. All methods use the same KG to ensure a fair comparison.

### 5.4 Evaluation Metrics

**Hits@1** measures question-level correctness, i.e., whether at least one predicted answer matches any gold answer entity.

**Precision**, **Recall**, and **F1** measure answer-set quality by evaluating exactness, completeness, and their balance over predicted and gold entities.

**KG-grounded Rate (KGR)** measures reasoning faithfulness as the proportion of questions answered via explicit KG reasoning rather than through fallback parametric generation.

### 5.5 Evaluation Protocol

This work uses strict exact-match evaluation: predicted answers must exactly match gold entities, without alias expansion or fuzzy matching. Hits@1, Precision, Recall, and F1 are all computed under this exact-match criterion. A question is counted as KG-grounded if the system constructs a complete relation chain, retrieves candidate entities from the KG, and outputs answers from those candidates via either the *Stop* or *Filter* action.

## 6 Experimental Results

### 6.1 Main Results

#### 6.1.1 Overall Improvements

Table 1 presents a comprehensive comparison of CoR and state-of-the-art baselines on WebQSP, CWQ, and QALD10 across two LLM backbones. Overall, CoR shows the most stable performance profile across datasets and backbones. It consistently leads on WebQSP and CWQ and remains competitive on QALD10. On QALD10, PoG attains the highest Hits@1 and KGR with DeepSeek-V3.1, while CoR remains close. QALD10 is based on Wikidata, where entity incompleteness appears less frequently, so the relative advantage of CoR is smaller than on WebQSP and CWQ.

In terms of accuracy (Hits@1, Recall, F1), CoR achieves clear gains on WebQSP and CWQ over ToG and PoG. The largest improvement appears in recall on WebQSP under GPT-4.1-mini. These gains are consistent with progressively stronger backtracking across methods and with global entity filtering, which avoids early elimination of valid candidates.

[4]https://query.wikidata.org/sparql

Table 1: Comparison of methods on WebQSP, CWQ (depth=4), and QALD10 under two LLM backbones (GPT-4.1-mini and DeepSeek-V3.1). Metrics include Hits@1, Recall (R), F1, and KG-grounded Rate (KGR).

| Method | WebQSP | | | | CWQ | | | | QALD10 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Hits@1 | R | F1 | KGR | Hits@1 | R | F1 | KGR | Hits@1 | R | F1 | KGR |
| **GPT-4.1-mini** | | | | | | | | | | | | |
| IO-Prompt | 62.1 | 43.7 | 47.7 | - | 36.0 | 30.8 | 32.1 | - | 44.8 | 40.7 | 41.5 | - |
| CoT-Prompt | 61.2 | 42.4 | 46.4 | - | 39.4 | 33.8 | 35.2 | - | 47.6 | 43.4 | 44.3 | - |
| ToG | 75.1 | 55.0 | 57.5 | 56.5 | 47.3 | 39.9 | 41.2 | 30.3 | 52.7 | 47.4 | 47.1 | 29.4 |
| PoG | 77.4 | 60.4 | 64.0 | 63.3 | 48.8 | 40.9 | 42.2 | 37.1 | 66.0 | 63.2 | 63.9 | 61.3 |
| CoR (ours) | **83.8** | **75.6** | **74.9** | **91.6** | **60.3** | **55.8** | **52.1** | **77.0** | **69.8** | **67.5** | **66.8** | **63.1** |
| **DeepSeek-V3.1** | | | | | | | | | | | | |
| IO-Prompt | 59.2 | 42.3 | 46.0 | - | 36.7 | 31.8 | 33.1 | - | 45.4 | 41.5 | 42.2 | - |
| CoT-Prompt | 59.3 | 42.0 | 45.7 | - | 39.6 | 34.1 | 35.5 | - | 48.3 | 43.9 | 44.8 | - |
| ToG | 74.6 | 53.5 | 56.3 | 56.4 | 45.8 | 35.2 | 37.3 | 28.6 | 51.4 | 46.7 | 46.5 | 32.7 |
| PoG | 76.7 | 61.3 | 64.6 | 62.8 | 48.3 | 39.5 | 41.4 | 35.8 | **68.5** | 63.8 | 64.2 | **64.9** |
| CoR (ours) | **80.2** | **71.0** | **70.3** | **88.5** | **57.9** | **52.8** | **50.6** | **69.7** | 68.3 | **66.0** | **65.3** | 63.1 |

Table 2: Effect of reasoning depth on CoR performance on CWQ (GPT-4.1-mini). Metrics include Hits@1, F1, and KG-grounded Rate (KGR).

| Depth | Hits@1 | F1 | KGR |
|---|---|---|---|
| 3 | 57.3 | 48.3 | 67.0 |
| 4 | **60.3** | **52.1** | **77.0** |
| 5 | 58.4 | 49.4 | 72.9 |

Table 3: Ablation Study on CWQ using GPT-4.1-mini. RB denotes Relation-level Backtracking, and GEF denotes global entity filtering.

| | Hits@1 | F1 |
|---|---|---|
| CoR | **60.3** | **52.0** |
| CoR w/o RB | 57.2 | 50.0 |
| CoR w/o GEF | 48.2 | 42.8 |

Table 4: Total tokens used by ToG, PoG, and CoR on WebQSP, CWQ (depth=4), and QALD10 with GPT-4.1-mini.

| Method | WebQSP | CWQ | QALD10 |
|---|---|---|---|
| ToG | 11,591,404 | 62,348,816 | 3,075,735 |
| PoG | 11,340,446 | 113,186,196 | 5,713,781 |
| CoR | **8,738,000** | **55,495,295** | **1,887,760** |

In terms of faithfulness (KGR), CoR also shows clear advantages on WebQSP and CWQ. CoR remains clearly ahead on both datasets under both backbones. On QALD10, CoR attains the highest KGR with GPT-4.1-mini, while PoG leads with DeepSeek-V3.1. This pattern is consistent with dataset characteristics. QALD10 is based on Wikidata, where entity incompleteness is less frequent, so the faithfulness gap is smaller than on WebQSP and CWQ.

Although PoG achieves performance close to CoR on QALD10, this comes with substantially higher inference cost. As shown in Figure 3 and Table 4, PoG requires more LLM calls and has the highest token usage overall.

#### 6.1.2 Impact of Reasoning Depth

Table 2 analyzes CoR under different maximum reasoning depths on CWQ with GPT-4.1-mini. Depth 4 achieves the best overall performance across all three metrics, indicating the most effective balance between exploration completeness and error control.

Compared with depth 4, depth 3 under-explores long reasoning chains and yields lower performance. Increasing to depth 5 slightly improves coverage over depth 3 but introduces additional noisy branches, so it remains below depth 4.

### 6.2 Ablation Study

To validate the contribution of each core component in CoR, this work conducts ablation studies by removing relation-level backtracking (RB) and global entity filtering (GEF) separately. Table 3 presents the results on CWQ.

**Impact of Relation-level Backtracking.** Removing relation-level backtracking leads to moderate decreases in both Hits@1 and F1 on CWQ (Hits@1: 60.3→57.2; F1: 52.0→50.0). This demonstrates that backtracking helps recover from exploration dead-ends, maintaining the ability to construct

Table 5: Case study on the question "Who inspired Obama?" (Gold answer: {Nipsey Russell, Reinhold Niebuhr, Saul Alinsky})

| Method | Extracted Answer | Reasoning Path | LLM Calls | Recall |
|---|---|---|---|---|
| **CoT** | Martin Luther King Jr., Rosa Parks, ... | None | 1 | 0/3 |
| **ToG** | Reinhold Niebuhr, Saul Alinsky | [Barack Obama, influenced.by, Reinhold Niebuhr]<br>[Barack Obama, influenced.by, Saul Alinsky]<br>[Barack Obama, parents, Ann Dunham] | 5 | 2/3 |
| **PoG** | **Nipsey Russell, Reinhold Niebuhr, Saul Alinsky** | [Barack Obama, influenced.by, Nipsey Russell]<br>[Barack Obama, influenced.by, Reinhold Niebuhr]<br>[Barack Obama, influenced.by, Saul Alinsky] | 7 | **3/3** |
| **CoR (ours)** | **Nipsey Russell, Reinhold Niebuhr, Saul Alinsky** | Barack Obama $\xrightarrow{\text{influenced.by}}$ (Nipsey Russell, Reinhold Niebuhr, Saul Alinsky) | **2** | **3/3** |

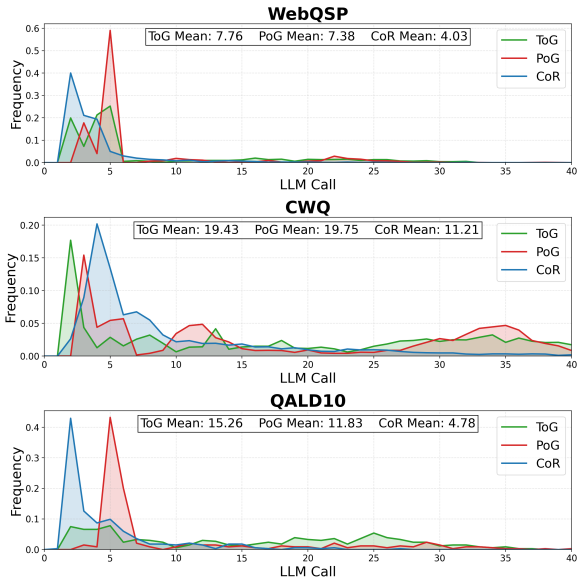



Figure 3: Efficiency comparison showing the distribution of LLM API calls per question for CoR, ToG, and PoG on WebQSP, CWQ, and QALD10 datasets. CoR demonstrates superior efficiency with most questions requiring fewer API calls.

valid reasoning paths in complex multi-hop settings.

**Impact of Global Entity Filtering.** Removing global entity filtering causes more substantial declines in both Hits@1 and F1 (Hits@1: 60.3→48.2; F1: 52.0→42.8). This shows that on CWQ, where questions involve multiple constraints and longer reasoning chains, deferring constraint validation until complete path construction is critical for answer quality.

### 6.3 Efficiency Study

To evaluate computational efficiency, this work first examines the distribution of LLM API calls per question on WebQSP and CWQ. Figure 3 shows that CoR is concentrated in the low-call region, which indicates that most questions are solved with fewer interactions. By comparison, PoG has a longer high-call tail and ToG shows a flatter distribution. CoR also has the lowest average LLM calls across datasets, with 4.03 on WebQSP, 11.21 on CWQ, and 4.78 on QALD10.

This work then reports total token consumption on WebQSP, CWQ, and QALD10 with GPT-4.1-mini in Table 4. CoR uses the fewest total tokens across all three datasets. This result is consistent with the call-distribution trend and reflects the benefit of relation-centric exploration, which reduces repeated entity-level ranking at each step.

### 6.4 Case Study

Table 5 presents a comparative analysis of different methods on the question "Who inspired Obama?"

CoT relies entirely on parametric knowledge, generating semantically plausible but factually incorrect answers (Martin Luther King Jr., Rosa Parks). This demonstrates the necessity of grounding LLM reasoning in external knowledge graphs.

ToG successfully retrieves two correct answers through entity-centric exploration but misses Nipsey Russell due to premature entity pruning during beam search. The method also explores an irrelevant path "[Barack Obama, parents, Ann Dunham]", highlighting inefficient exploration under beam-width constraints. PoG achieves full recall through self-correcting planning but requires 7 LLM API calls, resulting in higher computational cost.

CoR achieves full recall with only 2 LLM API calls by constructing a single, unified relation chain "Barack Obama $\xrightarrow{\text{influenced.by}}$ (Nipsey Russell, Rein-

hold Niebuhr, Saul Alinsky)" that simultaneously reaches all answer entities. This demonstrates that relation-centric exploration achieves superior performance across accuracy, efficiency, and explainability.

## 7 Conclusion

This paper presents Chain-of-Relations (CoR), a framework that shifts from entity-centric to relation-centric exploration for knowledge graph question answering. By constructing reasoning paths through relation chains and postponing constraint validation until complete relation chain construction, CoR addresses entity incompleteness vulnerability and premature entity pruning issues. Experiments demonstrate substantial improvements in accuracy, faithfulness, and efficiency over strong baselines, establishing relation-centric exploration as an effective paradigm for faithful and efficient knowledge graph reasoning.

## Limitations

While CoR demonstrates significant improvements, several limitations require discussion. First, although CoR achieves faithful reasoning through explicit KG traversal, constraint validation still relies on LLM parametric knowledge rather than structured KG traversal. The LLM identifies and applies question constraints based on its internal knowledge, which may introduce inconsistencies when conflicting with KG facts. Second, despite improved efficiency, CoR still requires multiple LLM calls, which may challenge large-scale or real-time applications. These limitations suggest promising directions for future work to refine and extend the relation-centric exploration paradigm.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (62472461), and the Guangdong Basic and Applied Basic Research Foundation (2025A1515010129).

## References

Agnes Axelsson and Gabriel Skantze. 2023. Using large language models for zero-shot natural language generation from knowledge graphs. In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 39–54.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 78–106.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. *Advances in Neural Information Processing Systems*, 37:37665–37691.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. 2024. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. *arXiv preprint arXiv:2402.11163*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11196–11215.

Ben Liu, Jihai Zhang, Fangquan Lin, Cheng Yang, Min Peng, and Wotao Yin. 2025. Symagent: A neural-symbolic self-learning agent framework for complex reasoning over knowledge graphs. In *Proceedings of the ACM on Web Conference 2025*, pages 98–108.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2025. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations*.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

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

Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. 2022. Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pages 229–234. IEEE.

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

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2024. Boosting language models reasoning with chain-of-knowledge prompting. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4958–4981.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38.

# A Prompt Templates

This appendix provides the exact prompt templates used in our experiments for reproducibility.

## A.1 Relation Pruning

```
# Task Definition
You are a KG relation-pruning expert.
Given a question and forward/backward relation
candidates under the current reasoning path,
select the most useful top-3 relations for
answer-oriented expansion.

# Instruction
1. Select no more than top-3 relations from
the provided candidates.
2. Use the SPARQL snippets as the source of
truth for direction and role semantics.
3. Prioritize relations that address
unresolved question constraints.
4. Down-rank redundant or near-duplicate
relations.
5. Unless explicitly required,
down-rank schema/meta relations (e.g.,
type.object.type).
6. Keep rationale short and specific.

# Output Format
Return exactly one JSON object. No markdown,
no extra text.
{"rational": "brief reasoning for selection",
"relation_by_relevance": ["relation_1",
"relation_2"]}
relation_by_relevance must be ordered from
most relevant to less relevant.
Include no more than 3 relations.

Q: {{question}}
{{relations}}
Answer:
```

## A.2 Reasoning

```
# Task Definition
You are a KG reasoning-path validator for main
exploration.
Given a question and a SPARQL-style reasoning
chain, evaluate the path and output the next
action.

# Instruction
1. Treat the chain as SPARQL-style triples
with strict direction semantics.
2. Make two-step judgement:
- Q1: completely correct / partially correct /
incorrect.
- Q2 (only when Q1 is partially correct): do
current TargetEntity candidates likely contain
final answer? yes / no.
3. Judge Q2 by candidate-space semantics (not
visible sample names):
- yes if candidates are already in the correct
answer space/type and final answer is likely
a subset (answer in candidates), so only
constraint filtering is needed.
- no if candidates are still outside the
required answer space/type, so further
exploration is needed.
4. Map judgement to decision:
- completely correct -> stop
- incorrect -> backtrack
- partially correct + yes -> filter
- partially correct + no -> forward
5. Always output three keys: rationale,
decision, answer.
6. decision must be one of forward,
backtrack, filter, stop, and consistent with
step 4.
7. If decision is stop, answer must be a
non-empty list (count questions may return
aggregated value).
8. If decision is filter, forward, or
backtrack, answer must be an empty list.
9. Do not assess data reliability; only judge
path-question semantic alignment.

# Output Format (strict JSON only)
Return exactly one JSON object. No markdown,
no extra text.
{"rationale": "brief reasoning logic for
path validity and completeness", "decision":
"forward|backtrack|filter|stop", "answer":
["answer_1", "answer_2"]}

Q: {{question}}
Reasoning Chain (SPARQL-style):
{{reasoning_path}}
Answer:
```

## A.3 Filtering

```
# Task Definition
You are a KG answer filter.
Given a question and a reasoning chain whose
candidate set already likely contains the
final answer, select all valid answers from
the candidate TargetEntity list, or return an
aggregated value when the question asks for
aggregation (e.g., count).

# Instruction
1. Use only question semantics to filter
candidates; do not add external entities.
- Exception: for aggregation/count questions,
return the aggregated value as a string (e.g.,
"4") rather than entity names.
2. Return all valid candidates if multiple
answers are correct.
3. answer must be non-empty.
4. Keep your analysis concise.

# Output Format (strict JSON only)
Return exactly one JSON object. No markdown,
no extra text.
{"rationale": "brief filtering rationale",
"answer": ["answer_1", "answer_2"]}
answer must be a JSON string list.
answer must be non-empty.
For non-aggregation questions, keep answers
exactly from TargetEntity candidates.
For aggregation/count questions, return
aggregated values as strings.

Q: {{question}}
Reasoning Chain (SPARQL-style):
{{reasoning_path}}
Answer:
```

### A.4 Generate Directly

```
# Task Definition
Answer the question based on your knowledge.

# Instruction
1. Provide a brief reasoning for your answer.
2. Keep the rationale concise and
question-focused.
3. Return one or more answers as strings in a
JSON list.
4. If uncertain or no answer is available,
return an empty list.

# Output Format (strict JSON only)
Return exactly one JSON object. No markdown,
no extra text.
{"rationale": "brief reasoning", "answer":
["answer_1", "answer_2"]}
answer must be a JSON string list.
If no answer, return "answer": [].

Q: {{question}}
Answer:
```