Beyond Chains: Bridging Large Language Models and Knowledge Bases in Complex Question Answering

Yihua Zhu[1,3], **Qianying Liu**[3], **Akiko Aizawa**[2,3], **Hidetoshi Shimodaira**[1,4]
[1]Kyoto University
[2]University of Tokyo
[3]NII LLMC
[4]RIKEN AIP
zhu.yihua.22h@st.kyoto-u.ac.jp, ying@nii.ac.jp, aizawa@nii.ac.jp, shimo@i.kyoto-u.ac.jp

Abstract

Knowledge Base Question Answering (KBQA) aims to answer natural language questions using structured knowledge from KBs. While LLM-only approaches offer generalization, they suffer from outdated knowledge, hallucinations, and lack of transparency. Chain-based KG-RAG methods address these issues by incorporating external KBs, but are limited to simple chain-structured questions due to the absence of planning and logical structuring. Inspired by semantic parsing methods, we propose **PDRR**: a four-stage framework consisting of **P**redict, **D**ecompose, **R**etrieve, and **R**eason. Our method first **predicts** the question type and **decomposes** the question into structured triples. Then **retrieves** relevant information from KBs and guides the LLM as an agent to **reason** over and complete the decomposed triples. Experimental results show that our proposed KBQA model, PDRR, consistently outperforms existing methods across different LLM backbones and achieves superior performance on various types of questions.

Code — https://github.com/YihuaZhu111/PDRR
Extended version — https://arxiv.org/abs/2505.14099

Introduction

Knowledge bases (KBs) offer rich, structured repositories of world knowledge, where facts are organized as (subject, relation, object) triples. Large-scale KBs such as Freebase (Bollacker et al. 2008), DBpedia (Lehmann et al. 2015), Wikidata (Pellissier Tanon et al. 2016), and YAGO (Suchanek, Kasneci, and Weikum 2007) provide useful resources for downstream applications. Knowledge base question answering (KBQA) leverages this structured information to translate natural language question queries into precise, verifiable answers, which poses challenges since it requires accurate multi-hop reasoning. This capability is essential for domains that require accurate and verifiable retrieval of facts.

In response to the costly human effort required to annotate training data, recent studies have turned to leveraging large language models (LLMs) based methods for KBQA, for example, IO (Brown et al. 2020), CoT (Wei et al. 2022), and SC (Wang et al. 2022). These methods harness

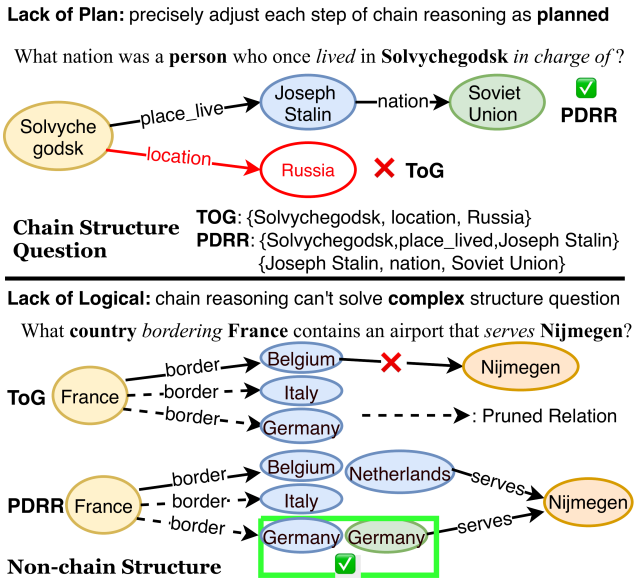

Figure 1: Drawbacks of ToG (a chain-based KG-RAG approach). ToG and similar chain-based KG-RAG methods lack a planning module for explicit reasoning control and are limited to chain-type questions due to their insufficient logical structuring. Our PDRR framework resolves both issues.

broad, pre-trained knowledge of LLMs and exhibit strong generalization across datasets. Nonetheless, LLM's knowledge may not reflect recent or domain-specific facts; they could also exhibit hallucinations, which is difficult to audit or verify, leading to unfaithful and unsupported response. To address these limitations, knowledge-graph retrieval-augmented generation (KG-RAG) methods further extend LLMs with retrieved structural knowledge. Various studies such as ToG (Sun et al. 2023), GoG (Xu et al. 2024), and CoK (Li et al. 2023) leverage chain-based KG-RAG approaches that performs reasoning over retrieved chains of KG facts. Specifically, they infer intermediate "bridge" entities at each hop, which repeatedly fetches triplets linked to the current entities from the KG, and uses the LLM to choose which relation and entity to retain for the next

step. These chain-based KG-RAG methods ground multi-step reasoning in explicit KG paths, thereby mitigating outdated knowledge, reducing hallucinations through factual grounding and rendering each inference step to be transparent.

Despite their strengths, existing chain-based KG-RAG methods exhibit two principal limitations as shown in Figure 1. First, they reason over the question in a holistic manner and lack a mechanism to structure the inference into targeted sub-tasks. In the upper half "Lack of Plan" example, one should first identify the notable person and then determine the nation they governed. Instead, ToG treats the query as an indivisible whole, leading it to select the superficially matching *location.containedby* relation rather than the correct *people.place_live.person*.

Second, chain-based approaches cannot handle richer logical structures beyond simple, linear hops, such as conjunction questions that require the intersection of multiple constraint sets. Logic structure refers to the structural form of reasoning required to answer a question. In the lower half "Lack of Logical" example, the answer hinges on intersecting (a) countries bordering France and (b) countries with airports serving Nijmegen. ToG's linear, single-path search prunes away additional candidates after the first hop, making it incapable of resolving the conjunction and thus yielding an incorrect result.

Motivated by these challenges, we propose the **P**redict–**D**ecompose–**R**etrieve–**R**eason (PDRR) framework. Inspired by pre-LLM semantic-parsing (SP-based) methods (Lan et al. 2022) that translate questions into executable KB logic forms, PDRR introduces a planning module (Predict, Decompose) for structuring multi-step inference, followed by a retrieval and reasoning module (Retrieve, Reason) that grounds the plan in KB facts and guides the LLM to execute it step by step, effectively simulating logical form execution over KGs. Finally, the question answering module utilizes the reasoning triples provided by the previous step to generate an answer.

First, our Predict stage classifies each query by type (i.e., chain or parallel) to determine an appropriate reasoning strategy and plan structure before retrieval begins. Next, the Decompose stage converts the question into a set of partial KG triples that reflect the plan, which breaks the overall query into manageable inference units, ensuring each reasoning step is focused and auditable. The Retrieve stage issues targeted KB lookups to fill in missing triple elements, which grounds the planned inference steps in factual knowledge. Finally, the Reason stage leverages the LLM to verify and complete each triple in accordance with the plan. By unifying logical-form-style planning with LLM-driven execution, PDRR supports a variety of reasoning patterns while preserving transparency.

We evaluate PDRR on four standard KBQA benchmarks: CWQ (Talmor and Berant 2018), WebQSP (Berant et al. 2013), Simple Question(Bordes et al. 2015), and GrailQA(Gu et al. 2021). Experimental results show that our proposed KBQA model, PDRR, consistently outperforms existing methods across different LLM backbones and achieves superior performance on various types of questions, demonstrating the effectiveness of PDRR's explicit planning module and the precise control over retrieval and reasoning modules.

Related Work

Semantic Parsing-Based Methods Before LLMs, KBQA was dominated by SP-based approaches, which generate logical forms to query structured KBs. CBR-KBQA (Das et al. 2021) combines a non-parametric memory of question–logical form pairs with a parametric retriever to guide logical form generation. TemplateQA (Zheng et al. 2018) bypasses complex parsing by matching questions to a large set of binary templates. SPARQA (Sun et al. 2020) uses a skeleton grammar and a BERT-based coarse-to-fine parser to handle complex questions. While interpretable, SP-based methods are limited by incomplete KBs and the need for additional model training.

LLM Retrieval Augmented Methods The emergence of LLMs has led to LLM-only approaches that require no additional training and leverage internalized general knowledge to mitigate KB incompleteness, such as IO (Brown et al. 2020), CoT (Wei et al. 2022), and SC (Wang et al. 2022). Despite their simplicity, these methods struggle with outdated knowledge, limited reasoning transparency, and hallucinations. To mitigate these issues, LLM+KG approaches have been developed. ToG (Sun et al. 2023) uses an LLM to guide reasoning over KGs by dynamically selecting relations and entities. CoK (Li et al. 2023) adaptively selects KGs and performs stepwise retrieval. GoG (Xu et al. 2024) augments KG coverage by letting LLMs infer missing links.

However, most chain-based KG-RAG methods lack explicit planning and are confined to chain reasoning, limiting reasoning control. Recent methods like chatKBQA (Luo et al. 2023a) and RoG (Luo et al. 2023b) introduce planning by fine-tuning LLMs to generate logic forms or reasoning paths, but they remain training-intensive. PoG (Chen et al. 2024) and DoG (Li et al. 2025) plan well but remain limited on non-chain complex questions. To address this, we propose PDRR: a training-free framework with explicit planning, enabling precise reasoning control and handling of complex question structures.

Preliminary

In this section, we first introduce Knowledge Graphs (KGs). Then, we use notations of KGs to describe reasoning triples and Knowledge Base Question Answering (KBQA).

Knowledge Graphs A KG $\mathcal{G}$ consists of factual triples $(e^h, r, e^t) \in \mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where e^h, e^t represent head and tail entities, and $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively.

Reasoning Triples The Reasoning Triples $\mathcal{T}^q$ denote the set of triples retrieved from the KG $\mathcal{G}$ to answer the question q, formally defined as:

$$\mathcal{T}^q = \left\{ \mathcal{T}_n^q = \left(e_n^h, r_n, e_n^t\right) \mid n = 1, 2, \ldots, m \right\} \subseteq \mathcal{G}$$

Here, $\mathcal{T}_n^q$ denotes the n-th reasoning triple.

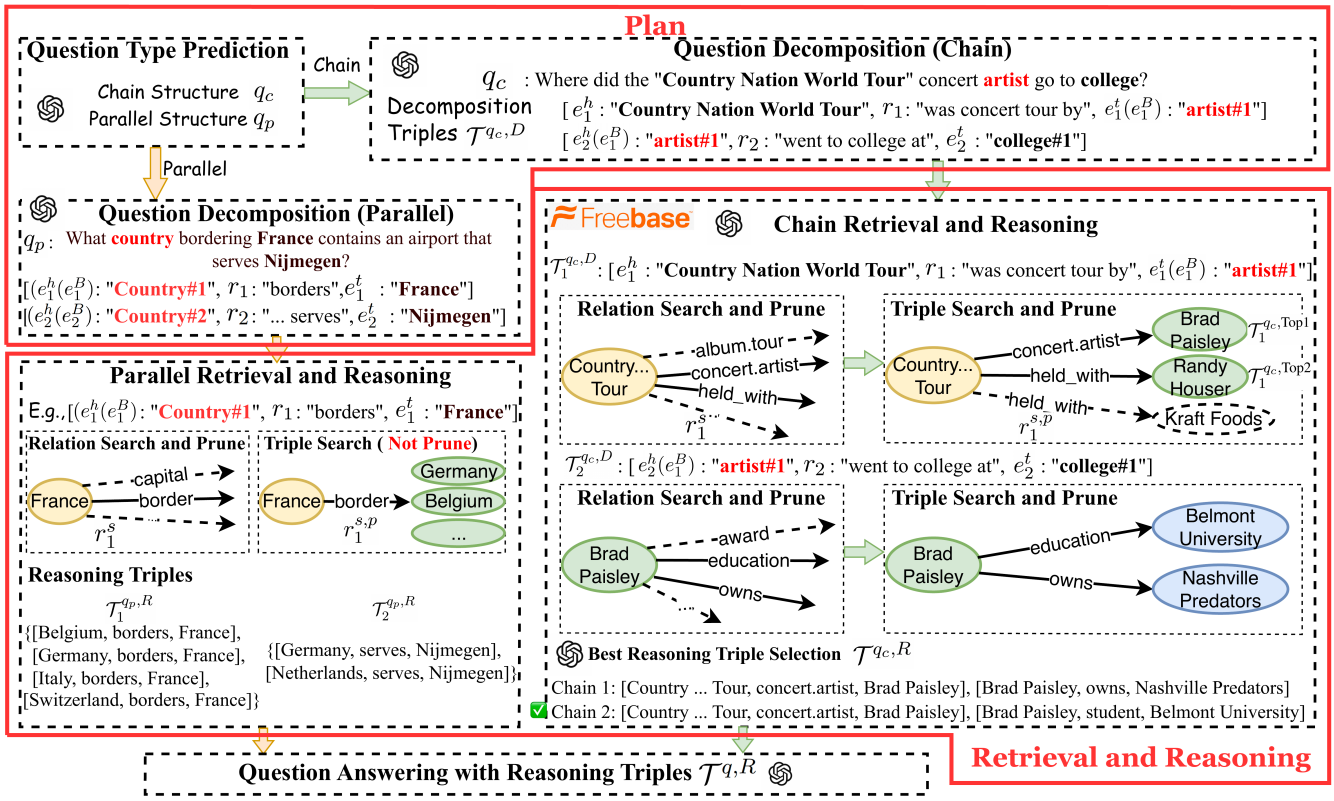

Figure 2: The framework of the PDRR method. The process follows the Predict-Decompose-Retrieve-Reason pipeline. Dashed lines and circles indicate pruned components with low relevance to the specific decomposed triple $\mathcal{T}_n^{q,D}$. Labeled *entity#index* (e.g., *artist#1*) elements denote key bridge entities e_n^B that are essential for reasoning.

Knowledge Base Question Answering KBQA involves reasoning over a KG to answer natural language questions. Formally, given a question q and a knowledge graph $\mathcal{G}$, the objective is to learn a function f that returns answers $a \in \mathcal{A}_q$ based on the information in $\mathcal{G}$, i.e., $a = f(q, \mathcal{G})$. In line with previous studies (Sun, Bedrax-Weiss, and Cohen 2019; Jiang et al. 2022), we assume that the entities $e_q \in \mathcal{Q}_q$ mentioned in the question and the gold answers $a \in \mathcal{A}_q$ are annotated and linked to entities in the KG, such that $\mathcal{Q}_q, \mathcal{A}_q \subseteq \mathcal{E}$, where $\mathcal{E}$ denotes the entity set of $\mathcal{G}$.

Approach

We propose a training-free method to address the lack of planning and logical structure handling in existing chain-based KG-RAG approaches. Our approach consists of three main modules: Plan, Retrieval and Reasoning, and Question Answering. The Plan module predicts the question type (i.e., chain or parallel) and decomposes the question accordingly. The Retrieval and Reasoning module retrieves the KB for relevant facts and employs the LLM to reason over and complete the decomposed triples. Finally, the Question Answering Module generates the final answer. Figure 2 illustrates the overall framework of our approach.

Plan Module

Our Plan module resembles logical form generation in SP-based methods. It first predicts the question type (i.e., chain or parallel) and then decomposes the question into decomposition triples, which serve a similar role to logical forms by guiding subsequent retrieval and reasoning. The motivation is to support not only chain-structured questions but also more complex reasoning patterns, while enabling precise downstream control through the generated decomposition triples.

Question Type Prediction Given a question q, we first use few-shot learning with LLMs predict its structural type, which we classify into two main categories: chain structure q_c and parallel structure q_p. The chain structure is the most common type in chain-based KG-RAG methods. Its reasoning process is sequential, where each step depends on the bridge entity e_n^B identified in each step of reasoning triple $\mathcal{T}_n^q$. Therefore, the reasoning steps are interdependent and must be performed in order. In contrast, the parallel structure comprises multiple logically independent sub-steps that can be executed concurrently. More detail refers to section: Discussion of Chain and Parallel Reasoning.

Question Decomposition Based on the predicted question type, question q is decomposed into KG-style triples $\mathcal{T}^{q,D}$, providing precise control for downstream processing.

For chain-structured questions q_c, we identify all bridge entities e_n^B (denoted as *entity#index*) to distinguish them in multi-hop reasoning. Each decomposition triple $\mathcal{T}n^{q_c,D} = (e_n^h, r_n, e_n^t)$ connects entities via relation r_n, with adjacent triples linked through e_n^B such that $e_{n+1}^h = e_n^t = e_n^B$, forming a reasoning chain. For example, in Figure 2, the bridge entity *artist#1* connects two steps: identifying the *artist#1* of Country Nation World Tour, and determining the college that the *artist#1* attended.

For parallel-structured questions, we similarly identify bridge entities and construct KG-style triples along independent reasoning paths. Unlike chain structures, these steps are mutually independent and can be executed in parallel. For example, in Figure 2, the reasoning involves two conditions: identifying *country#1* bordering France and *country#2* with an airport serving Nijmegen, as reflected in the decomposition triples.

Chain Retrieval and Reasoning Module

In Retrieval and Reasoning module, we adopt different reasoning strategies based on the identified question type. Similar to how SP-based methods use logic forms to retrieve information from KBs, we leverages KB knowledge and employs the LLM as an agent to complete missing information in the decomposition triples $\mathcal{T}^{q,D}$ generated by the plan module. In this module, if no relevant information can be retrieved from the KB, the system directly generates the answer using the CoT approach.

Given a chain-type question q_c and its decomposition triples $\mathcal{T}^{q_c,D}$, we sequentially complete them by identifying bridge entities step by step, enabling accurate construction of reasoning triples and final answer derivation.

Complete First Decomposition Triples We first extract $\mathcal{T}_1^{q_c,D} = (e_1^h, r_1, e_1^t(e_1^B))$ and retrieve the bridge entity e_1^B using the method illustrated in Figure 2.

- **Relation Search and Prune** We first apply SPARQL fuzzy matching to obtain the Freebase entity ID of the non-bridge entity in the triple thought its string. In the **Search** phase, we query the KB with this entity ID to obtain all connected relations r_1^s. In the case of Figure 2, we first identify the entity ID of e_1^h(*Country Nation World Tour*), and then retrieve all connected relations r_1^s.

  We then **prune** the retrieved relation set r_1^s. Unlike ToG (Sun et al. 2023), which ranks relations $r_{1,i}^s$ by their similarity $Sim(r_{1,i}^s, q)$ to the entire question q and may overlook local information, we rank them by their similarity $Sim(r_{1,i}^s, \mathcal{T}_1^{q_c,D})$ to the specific decomposition triple $\mathcal{T}1^{q_c,D}$. This yields the pruned set $r_1^{s,p}$. This helps avoid errors such as the one in Figure 1, where ToG selects *contained by*, which aligns with the global semantics of "`What nation`" instead of the correct local relation *people.place_live.person*.

  By aligning pruning with the decomposition triple, our approach ensures precise control and preserves step-level semantics. As shown in the Figure 2, we obtain a pruned relation set $r_1^{s,p} = \{$*concert_tour.artist*, *event.held_with*$\}$, which is most similar to the decomposition triple $\mathcal{T}_1^{q_c,D}$.

- **Triple Search and Prune** Given ID of e_1^h and the pruned relation set $r_1^{s,p}$, we use SPARQL to **search** all tail entities (bridge entities) $e_1^t(e_1^B)$ to construct the candidate triple set $\mathcal{T}_1^{q_c,\text{Cand}}$. In the Triple **Prune** stage, instead of pruning only entities as in ToG, which inherits the same limitations as its relation pruning, we rank candidate triples $\mathcal{T}_1^{q_c,\text{Cand}_i}$ based on their similarity $Sim(\mathcal{T}_1^{q_c,\text{Cand}_i}, \mathcal{T}_1^{q_c,D})$ to the decomposition triple $\mathcal{T}_1^{q_c,D}$ and retain the top two: $\mathcal{T}_1^{q_c,\text{Top1}} = \{$*Country Nation World Tour*, *concert_tour.artist*, *Brad Paisley*$\}$, and $\mathcal{T}_1^{q_c,\text{Top2}}$.

Complete Rest Decomposition Triples Through Relation Search and Prune and Triple Search and Prune, we identify the top two bridge entities in first decompostion triple: $e_1^{B,\text{top1}}$ and $e_1^{B,\text{top2}}$. We then replace the bridge entity in next decompostion triple $\mathcal{T}_2^{q_c,D}$ with the bridge entities from the previous step and repeat the same procedure until all decomposition triples in $\mathcal{T}^{q_c,D}$ are completed.

Choose Best Reasoning Triples We apply beam search to retain the top-2 triples at each hop, preserving sufficient information. For example, for a 2-hop question, this yields 4 reasoning triples. The LLM then selects the most relevant triple:$\mathcal{T}^{q_c,R}$ for answering the question.

Parallel Retrieval and Reasoning Module

Parallel-structured questions q_p follow a non-sequential reasoning logic, where the decomposition triples $\mathcal{T}^{q_p,D}$ are mutually independent, allowing all retrieval and reasoning steps to be executed concurrently.

For example, given the first decomposition triple $\mathcal{T}_1^{q_p,D}$ in Figure 2, we perform relation search and pruning, followed by triple search, using the same procedure as in the chain setting. Unlike the chain case, we skip triple pruning, as all reasoning triples are needed for the final answer. This process results in a set of reasoning triples $\mathcal{T}_1^{q_p,R}$.

We repeat this process for all decomposition triples $\mathcal{T}_k^{q_p,D}$ to obtain the corresponding reasoning triples $\mathcal{T}_k^{q_p,R}$, where k indexes each decomposition triple.

Discussion of Chain and Parallel Reasoning

While our question type classification is based on reasoning strategy, another key criterion is the number of bridge entities that the model should memorize during the reasoning process. Chain-structured questions require exactly one bridge entity between each pair of triples to support step-by-step reasoning, whereas parallel-structured questions allow multiple bridge entities without such constraints.

When applying chain reasoning to parallel-structured questions, retaining all retrieved triples without pruning can still lead to correct answers. For example, in Figure 2, all *countries* (bridge entities) bordering France are considered and individually checked for containing an airport that serves Nijmegen, which can lead to the right answer. However, this significantly increases computational cost. Intro-

Question type	Question Example	Canonical Reasoning Logic	Expected Reasoning Type
Composition	Where did the "Country Nation World Tour" concert artist go to college?	First, identify the artist of the concert tour "Country Nation World Tour," and then determine which college this artist attended.	Chain
Conjunction	What country bordering France contains an airport that serves Nijmegen?	Find the intersection of two sets: (1) all countries that border France, and (2) all countries that contain an airport that serves Nijmegen.	Parallel
Comparative	Which of the countries bordering Mexico have an army size of less than 1050?	First, identify all countries bordering Mexico, and then select those with an army size of less than 1050.	Parallel
Superlative	What movies does taylor lautner play in and is the film was released earliest?	First, identify all films in which Taylor Lautner appeared, and then find the one that was released the earliest.	Parallel

Table 1: Examples of the four question types in the CWQ dataset, their corresponding canonical reasoning logic, and the expected reasoning type for each.

ducing pruning in chain reasoning mitigates this cost but risks discarding correct answers.

Therefore, parallel reasoning can be regarded as a complementary strategy to chain reasoning, particularly in cases with numerous bridge entities where computational efficiency becomes critical.

In practice, the logical structures or types of questions are not limited to just chain and parallel. In our work, we focus only on chain and parallel logic structures because a well-designed parallel structure that complements the chain structure, combined with the general knowledge and reasoning capabilities of LLMs, is sufficient to address the majority of KBQA questions. This also allows our method to remain training-free, avoiding additional computational costs.

Question Answering Module

Given the original question q, the decomposition triples $\mathcal{T}^{q,D}$, and the reasoning triples $\mathcal{T}^{q,R}$. The LLM is guided to answer the question q by leveraging the information in retrieved reasoning triples $\mathcal{T}^{q,R}$ in accordance with the reasoning logic encoded in $\mathcal{T}^{q,D}$.

Experiments

Experiment Setup

Dataset To evaluate performance beyond simple chain-structured QA, we evaluate the model on the CWQ (Talmor and Berant 2018) test set (3,531 questions), which includes four question types: composition (45%), conjunction (45%), comparative (5%), and superlative (5%). Table 1 shows examples, reasoning logic, and expected types. For additional validation, we evaluate the model on the test sets of WebQSP (Berant et al. 2013) (1,639 questions, CC License), SimpleQuestions (Bordes et al. 2015) (CC License), and GrailQA (Gu et al. 2021).

Evaluation Metrics Consistent with prior studies (Luo et al. 2023b; Sun et al. 2023), we adopt Hits@1 as the evaluation metric, which reflects the percentage of questions for which the top-1 rank predicted answer is correct.

Baseline We divide the baselines into three groups. The first group includes LLM-only methods including IO (Brown et al. 2020) and CoT (Wei et al. 2022), and our ablated variant PDR (Predict-Decompose-Reason), which removes the retrieve stage from PDRR. The second group incorporates external KBs and require additional training, including UniKBQA (Jiang et al. 2022), ROG (Luo et al. 2023b), CBR-KBQA (Das et al. 2021), SPARQA (Sun et al. 2020), and ChatKBQA (Luo et al. 2023a). The third group leverages external KBs without additional training, such as ToG (Sun et al. 2023), StructGPT (Jiang et al. 2023), and our model, PDRR.

Implementation For all intermediate steps, the max token length is 256, and 1024 for final answer generation. A temperature of 0.1 is used to reduce hallucinations and improve control. We adopt 5-shot prompting for answer generation and question type classification, and 3-shot for all other components.

Accuracy Result

Main Results In this section, we compare PDRR with various baselines across KGQA datasets as shown in Table 2. On the more challenging CWQ dataset, which features diverse question structures beyond simple chains, PDRR performs competitively with training-based methods and outperforms training-free ones like ToG by nearly 10% in accuracy.

By question type, PDRR significantly surpasses ToG on composition questions (66.2 vs. 49.9), highlighting the effectiveness of the Plan Module in guiding step-by-step reasoning. It also achieves higher accuracy on conjunction questions and remains competitive on comparative and superlative types, showing that augmenting the Retrieval and Reasoning Module with complementary parallel reasoning enhances performance on complex, diverse questions.

On the simpler WebQSP, SimpleQuestions, and GrailQA datasets, which contain only composition-type questions, ToG and PDRR achieve comparable performance on WebQSP, while PDRR consistently outperforms ToG on both GrailQA and SimpleQuestions, indicating stronger generalizability across both simple and complex cases.

Different Backbone Models To evaluate the robustness of PDRR, we test its performance on the CWQ dataset using different LLM backbones to assess its effectiveness beyond

Method	CWQ					WebQSP	SimpleQA	GrailQA
	All	Composition	Conjunction	Comparative	Superlative	All	All	All
Without KB Knowledge, Without Training								
IO	44.3	41.3	50.6	33.8	27.9	70.8	27.1	36.8
CoT	44.8	44.2	49.5	30.5	27.4	72.1	27.3	37.2
PDR (ours)	45.7	49.3	46.4	31.5	26.4	68.9	27.5	37.5
With KB Knowledge, With Training								
SPARQA $^{\alpha}$	31.6	-	-	-	-	-	-	-
UniKGQA $^{\alpha}$	51.2	-	-	-	-	77.2	-	-
RoG $^{\alpha}$	62.6	-	-	-	-	85.7	-	-
CBR-KBQA $^{\alpha}$	67.1	-	-	-	-	69.9	-	-
ChatKBQA $^{\alpha}$	82.7	-	-	-	-	83.2	-	-
With KB Knowledge, Without Training								
StructGPT $^{\alpha,\beta}$	-	-	-	-	-	72.6	-	-
PDRR(ours) $^{\beta}$	37.7	34.5	43.9	26.3	25.9	75.4	-	-
ToG	48.9	49.9	50.1	42.7	**37.1**	**80.2**	57.2	65.5
PDRR(ours)	59.6	**66.2**	59.1	38.5	34.5	79.2	**59.3**	**71.7**
PDRR(gold type)(ours)	**62.2**	65.5	**64.6**	**43.7**	36.6	-	-	-

Table 2: Hit@1 accuracy results with different baselines on KGQA datasets. **Bold** denotes the best performance among the training-free methods, and underline indicates the second-best. Results marked with superscripts α are taken directly from the original papers. Except for models marked with β, which use GPT-3.5-turbo, all other results are reproduced using GPT-4o as the backbone. Our proposed methods include PDR, PDRR, and PDRR (gold type). In the gold type setting (an ablation setup), ground-truth question types from CWQ are used instead of LLM predictions. Composition-type questions are handled with chain reasoning, while all others use parallel reasoning.

GPT-4o. Specifically, we compare CoT, ToG, PDRR, and PDRR (gt) across four models: GPT-3.5-turbo, DeepSeek-V3, and GPT-4o. Additionally, we evaluate CoT, PDRR, and PDRR (gt) on LLaMA-3.3B-Instruct. As shown in Table 3, PDRR consistently outperforms across all models. The performance gap is modest on GPT-3.5-turbo but becomes more significant on the stronger LLMs. These results confirm that PDRR is robust and not dependent on any specific language model.

Discussion of Question Structure

A core component of PDRR is the use of the Plan Module to predict the question structure type and apply corresponding decomposition and reasoning strategies. Thus, investigating this process is essential.

Question Type Prediction We analyze how different LLMs predict the structure type (chain or parallel) for various CWQ question types, as shown in Figure 3.

For composition-type questions, most LLMs correctly predict a chain structure, consistent with expectations. For conjunction-type questions, which are best handled by parallel reasoning, GPT-3.5-turbo and DeepSeek-V3 predict the correct structure in most cases, while Llama3.3-Instruct and GPT-4o achieve a much lower rate. Despite this, the results in Table 2 reveal that many misclassified conjunction questions still lead to correct answers, as they can be effectively addressed by either chain or parallel reasoning.

This phenomenon can be explained by the number of bridge entities: when only one is involved, both chain and parallel reasoning are generally effective; with more, parallel reasoning becomes notably more robust. This supports the discussion in Section: Discussion of Chain and Parallel Reasoning.

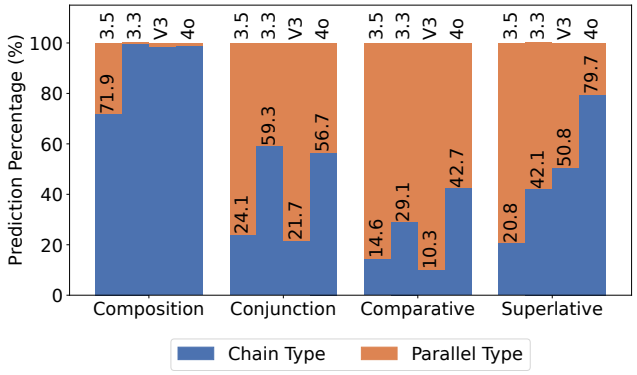

Figure 3: Predicted question structure types (chain or parallel) by different LLMs on various question types in the CWQ dataset. Blue indicates that the LLM predicts the question as a chain structure, while orange indicates a parallel structure. 3.5, 3.3, V3, and 4o in the figure denote GPT-3.5-turbo, Llama 3.3, DeepSeek V3, and GPT-4o, respectively.

Different Reasoning Strategies We evaluate the effectiveness of chain and parallel reasoning across four CWQ question types using Hits@1 accuracy, as shown in Figure 4. Chain reasoning performs significantly better than parallel

Method	CWQ				
	All	Compo	Conju	Compa	Super
GPT-3.5-turbo-0125					
CoT	36.8	33.8	43.0	26.8	20.8
ToG	30.9	30.1	35.6	16.4	15.8
PDRR	37.7	34.5	43.9	26.3	25.9
PDRR(gt)	**40.7**	**36.1**	**48.6**	**29.6**	**26.4**
Llama3.3-Instruct					
CoT	46.0	41.5	52.4	42.7	33.5
PDRR	54.1	54.9	57.7	36.2	38.1
PDRR(gt)	**58.7**	**56.4**	**65.0**	**46.5**	**39.1**
Deepseek-V3					
CoT	44.3	43.0	48.5	35.7	29.9
ToG	48.0	48.5	51.7	35.7	27.4
PDRR	56.0	**60.3**	57.0	41.3	30.5
PDRR(gt)	**57.1**	**60.3**	**57.5**	**43.7**	**42.6**
GPT-4o-2024-11-20					
CoT	44.8	44.2	49.5	30.5	27.4
ToG	48.9	49.9	50.1	42.7	**37.1**
PDRR	59.6	**66.2**	59.1	38.5	34.5
PDRR(gt)	**62.2**	65.5	**64.6**	**43.7**	36.6

Table 3: Performance of PDRR with different backbone models on overall accuracy and by question type in the CWQ dataset. 'gt' denotes the gold type setting.

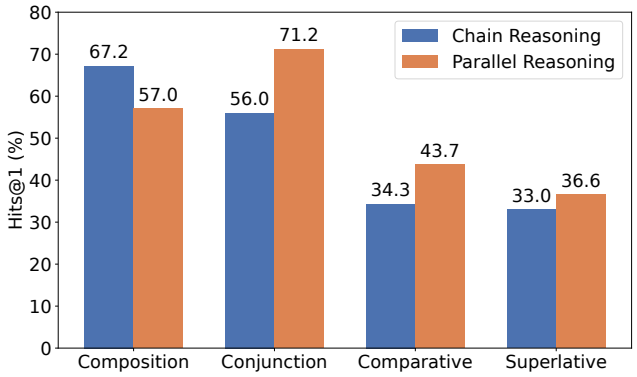

Figure 4: Hits@1 accuracy of different question types in the CWQ dataset under chain and parallel reasoning strategies. The evaluation includes 500 chain, 500 intersection, 213 comparative, and 197 superlative questions. GPT-4o is used as the LLM backbone.

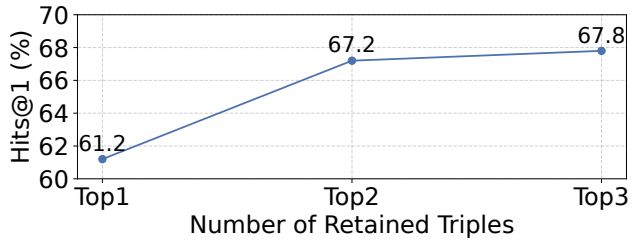

Figure 5: Hits@1 accuracy on the first 500 composition-type questions in CWQ using chain reasoning with different number of retained triples.

on composition-type questions (67.2% vs. 57.0%), while parallel reasoning clearly outperforms chain on conjunction-type questions (71.2% vs. 56.0%) and slightly outperforms it on comparative and superlative questions.

These results align with our hypothesis: composition-type questions favor chain reasoning, whereas the others benefit more from parallel reasoning. In contrast, methods like ToG, which rely solely on chain reasoning, underperform on non-chain questions. This highlights the need to adapt reasoning strategies to question type.

Ablation Study

Individual effects of the Retrieval and Reasoning Module We conducted experiments where this module was removed, relying only on the Plan Module and the final QA step to generate answers. The results are shown in Table 2 under "PDR (ours)", where the performance on CWQ is 45.7%. This already outperforms other prompting-based baselines such as IO (44.8%) and CoT (44.3%), demonstrating the effectiveness of the Plan Module. At the same time, the significant performance drop from the full PDRR (59.6%) to PDR (45.7%) confirms that the Retrieval and Reasoning Module plays a crucial role in the overall framework.

Number of Retained Triples Within Chain Reasoning We aim to explore the impact of the Number of Retained Triples within Chain Reasoning on the experimental results. As shown in Figure 5, increasing the number of retained triples from top-1 to top-2 significantly improves performance (61.2% to 67.2%). Although top-3 offers a minor additional gain (+0.6%), we adopt top-2 to balance accuracy and computational cost.

Conclusion

To address the limitation of chain-based KG-RAG methods, which are restricted to simple chain-structured questions due to the lack of planning and logical structuring. We propose PDRR, which first **predicts** the question type and **decomposes** the question into structured triples. Then **retrieves** relevant information from KBs and guides the LLM as an agent to **reason** over and complete the decomposed triples. Experimental results show that PDRR consistently outperforms existing methods across various LLM backbones, with up to a 10% gain on CWQ using GPT-4o. It also performs robustly across diverse question types.

Acknowledgments

This work was partially supported by JST SPRING JPMJSP2110 (YZ), JSPS KAKENHI 22H05106, 23H03355, and JST CREST JPMJCR21N3 (HS).

References

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In

Yarowsky, D.; Baldwin, T.; Korhonen, A.; Livescu, K.; and Bethard, S., eds., *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1533–1544. Seattle, Washington, USA: Association for Computational Linguistics.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 1247–1250.

Bordes, A.; Usunier, N.; Chopra, S.; and Weston, J. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, L.; Tong, P.; Jin, Z.; Sun, Y.; Ye, J.; and Xiong, H. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. *Advances in Neural Information Processing Systems*, 37: 37665–37691.

Das, R.; Zaheer, M.; Thai, D.; Godbole, A.; Perez, E.; Lee, J.-Y.; Tan, L.; Polymenakos, L.; and McCallum, A. 2021. Case-based reasoning for natural language queries over knowledge bases. *arXiv preprint arXiv:2104.08762*.

Gu, Y.; Kase, S.; Vanni, M.; Sadler, B.; Liang, P.; Yan, X.; and Su, Y. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the web conference 2021*, 3477–3488.

Jiang, J.; Zhou, K.; Dong, Z.; Ye, K.; Zhao, W. X.; and Wen, J.-R. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Jiang, J.; Zhou, K.; Zhao, W. X.; and Wen, J.-R. 2022. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *arXiv preprint arXiv:2212.00959*.

Lan, Y.; He, G.; Jiang, J.; Jiang, J.; Zhao, W. X.; and Wen, J.-R. 2022. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11): 11196–11215.

Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; Van Kleef, P.; Auer, S.; et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2): 167–195.

Li, K.; Zhang, T.; Wu, X.; Luo, H.; Glass, J.; and Meng, H. 2025. Decoding on graphs: Faithful and sound reasoning on knowledge graphs through generation of well-formed chains. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 24349–24364.

Li, X.; Zhao, R.; Chia, Y. K.; Ding, B.; Joty, S.; Poria, S.; and Bing, L. 2023. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *arXiv preprint arXiv:2305.13269*.

Luo, H.; Tang, Z.; Peng, S.; Guo, Y.; Zhang, W.; Ma, C.; Dong, G.; Song, M.; Lin, W.; Zhu, Y.; et al. 2023a. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. *arXiv preprint arXiv:2310.08975*.

Luo, L.; Li, Y.-F.; Haffari, G.; and Pan, S. 2023b. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Pellissier Tanon, T.; Vrandečić, D.; Schaffert, S.; Steiner, T.; and Pintscher, L. 2016. From freebase to wikidata: The great migration. In *Proceedings of the 25th international conference on world wide web*, 1419–1428.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, 697–706.

Sun, H.; Bedrax-Weiss, T.; and Cohen, W. W. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*.

Sun, J.; Xu, C.; Tang, L.; Wang, S.; Lin, C.; Gong, Y.; Ni, L. M.; Shum, H.-Y.; and Guo, J. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.

Sun, Y.; Zhang, L.; Cheng, G.; and Qu, Y. 2020. SPARQA: skeleton-based semantic parsing for complex questions over knowledge bases. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 8952–8959.

Talmor, A.; and Berant, J. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In Walker, M.; Ji, H.; and Stent, A., eds., *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 641–651. New Orleans, Louisiana: Association for Computational Linguistics.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xu, Y.; He, S.; Chen, J.; Wang, Z.; Song, Y.; Tong, H.; Liu, G.; Liu, K.; and Zhao, J. 2024. Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering. *arXiv preprint arXiv:2404.14741*.

Zheng, W.; Yu, J. X.; Zou, L.; and Cheng, H. 2018. Question answering over knowledge graphs: question understanding via template decomposition. *Proceedings of the VLDB Endowment*, 11(11): 1373–1386.