From Query to Logic: Ontology-Driven Multi-Hop Reasoning in LLMs

Haonan Bian[1], Yutao Qi[1,*], Rui Yang[1], Yuanxi Che[1],
Jiaqian Wang[1], Heming Xia[2], Ranran Zhen[3]
[1]Xidian University [2]The Hong Kong Polytechnic University [3]Soochow University
[*]Corresponding author: ytqi@xidian.edu.cn

Abstract

Large Language Models (LLMs), despite their success in question answering, exhibit limitations in complex multi-hop question answering (MQA) tasks that necessitate non-linear, structured reasoning. This limitation stems from their inability to adequately capture deep conceptual relationships between entities. To overcome this challenge, we present **ORACLE** (**O**ntology-driven **R**easoning **A**nd **C**hain for **L**ogical **E**lucidation), a training-free framework that combines LLMs' generative capabilities with the structural benefits of knowledge graphs. Our approach operates through three stages: (1) dynamic construction of question-specific knowledge ontologies using LLMs, (2) transformation of these ontologies into First-Order Logic (FOL) reasoning chains, and (3) systematic decomposition of the original query into logically coherent sub-questions. Extensive experiments across a diverse set of models and standard MQA benchmarks demonstrate that our framework achieves competitive performance while producing more interpretable reasoning chains.

1 Introduction

Large Language Models have demonstrated significant success in knowledge-based question answering (Brown et al., 2020; DeepSeek-AI, 2025; Yang et al., 2025). However, they continue to face challenges in MQA tasks (Wang et al., 2024; Shao et al., 2023a), which require integrating and reasoning over multiple, discrete sources of information. A significant challenge, as highlighted in Ju et al. (2024), is that LLMs tend to rely on guessing derived from training data rather than actually reasoning in MQA tasks. Therefore, the key lies in advancing beyond the generation of factually correct chains toward empowering LLMs to uncover and leverage deeper conceptual relationships between entities—a capability essential for true multi-hop understanding.

Recent research on MQA has predominantly focused on two paradigms: prompting strategies and Retrieval-Augmented Generation (RAG). Standard RAG approaches often falter in MQA as they struggle to retrieve all necessary information fragments in a single pass. Iterative retrieval methods like ReAct (Yao et al., 2023) and EfficientRAG (Zhuang et al., 2024) address this by progressively refining the retrieved results. Specifically, EfficientRAG employs a smaller model as an evaluator and retrieval generator to streamline the process without constant reliance on a large model. Another line of methods, exemplified by PAR-RAG (Zhang et al., 2025), centers on upfront problem decomposition to facilitate more globally coherent solution planning. Concurrently, researchers have proposed leveraging Knowledge Graphs (KGs) to represent the MQA reasoning process, capitalizing on their structured nature. Among these studies, LPKG (Wang et al., 2024) utilizes inherent patterns within a KG to guide the LLM's decomposition and planning, while ROG (Luo et al., 2024a) uses reasoning paths from KG subgraphs to direct retrieval and problem-solving.

While these KG-based methods show promise, we argue that relying on predefined structural paths is insufficient. A potential limitation of existing LLM-based approaches is their tendency to focus on surface entity mentions, which may lead to overlooking the underlying attributes and inter-relations necessary for compositional reasoning. We posit that the essence of complex reasoning lies not only in the relationships between entities but also in the "concepts" they belong to and the hierarchical relationships between these concepts. To this end, we introduce **ORACLE**, a training-free MQA framework centered on a *Question-centric Knowledge Graph Ontology*. Instead of using static KG paths or requiring model fine-tuning, our framework dynamically constructs a bespoke ontology for each question using a powerful LLM. This ontology pro-

Findings of the Association for Computational Linguistics: ACL 2026, pages 14035–14051

vides a structured semantic scaffold, capturing the core entities, their interrelations, and underlying conceptual hierarchies, thereby guiding the LLM's reasoning process.

ORACLE operates in three sequential stages: (1) ontology construction, (2) First-Order Logic (FOL) formulation, and (3) sub-question decomposition. First, the LLM dynamically constructs a question-centric knowledge ontology. This ontology delineates the key entities within the question and their underlying conceptual relationships, establishing a structured foundation for the reasoning process. Subsequently, this ontology facilitates the translation of the original question into a formal FOL reasoning chain. This logical structure makes the required inferential steps explicit. In the final stage, guided by both the knowledge ontology and the FOL chain, the LLM systematically decomposes the complex initial query into a sequence of simpler, logically coherent sub-questions.

To sum up, our key contributions are as follows:

- We apply ontology theory from knowledge engineering to guide LLM reasoning. By dynamically constructing a *Question-centric Knowledge Ontology*, we enhance sub-problem decomposition for MQA tasks.
- We propose ORACLE, a training-free MQA framework that integrates ontological reasoning with FOL, achieving competitive performance on standard benchmarks.
- Our analysis demonstrates that ORACLE generates more interpretable reasoning chains than path-pattern approaches, providing new insights into LLM MQA reasoning.

2 Related Work

Multi-Hop Question Answering Early MQA research relied on Graph Neural Networks (Fang et al., 2020; Zhang et al., 2022; Li et al., 2022), though these methods were often limited by dataset-specific training. The advent of LLMs shifted the paradigm toward in-context learning (Wei et al., 2022) and Retrieval-Augmented Generation (RAG), with recent works optimizing retrieval through iterative feedback (Yao et al., 2023; Zhuang et al., 2024). Another prominent direction is question decomposition, where approaches like PAR-RAG (Zhang et al., 2025) break down complex questions into sub-plans, and LPKG (Wang et al., 2024) learns decomposition strategies from knowledge graphs (KGs). Our work advances this research on question decomposition. Instead of learning patterns from a static KG, we leverage knowledge representation principles to dynamically generate an ontology that structures the question into a logical chain.

KG-enhanced LLMs Knowledge Graphs (KGs) are widely used to provide structured, factual grounding for LLMs. One prominent line of research integrates KGs directly into the LLM inference process, for instance by augmenting context with retrieved triples (TOG (Sun et al., 2024)), generating relation-aware plans to mitigate hallucination (ROG (Luo et al., 2024a)), or constraining the decoding process with KG paths to ensure faithful reasoning (GCR (Luo et al., 2024b)). Another direction utilizes KGs to synthesize high-quality reasoning data for model fine-tuning, as demonstrated by MedReason (Wu et al., 2025) and OntoTune (Liu et al., 2025). While our work is conceptually aligned with methods like LPKG (Wang et al., 2024) that use KG structures for task decomposition, our key distinction is the dynamic use of KG schema. Rather than relying on static KG patterns, our framework constructs a question-centric ontology on-the-fly to guide the LLM's reasoning process.

First-Order Logic (FOL) Integrating LLMs with First-Order Logic (FOL) generally involves translating language for symbolic reasoners (Ye et al., 2023; Pan et al., 2023) or enhancing intrinsic reasoning via fine-tuning (Xu et al., 2024; Morishita et al., 2024). However, these methods often struggle to capture complex real-world nuances beyond formal logic. In Knowledge Graphs, Complex Query Answering (CQA) uses FOL embeddings for reasoning (Ren et al., 2020), yet these execution-centric models rely on fixed query structures and lack the interface for autonomous formula construction, necessitating "abductive reasoning" (Bai et al., 2025). We propose a **dynamic query planner** that, unlike traditional CQA, leverages LLMs to dynamically construct ontologies and FOL chains. By synergizing logical rules with structured knowledge, our approach effectively handles complex queries without predefined schemas.

3 Preliminary

This section introduces the core terminology used in this paper.

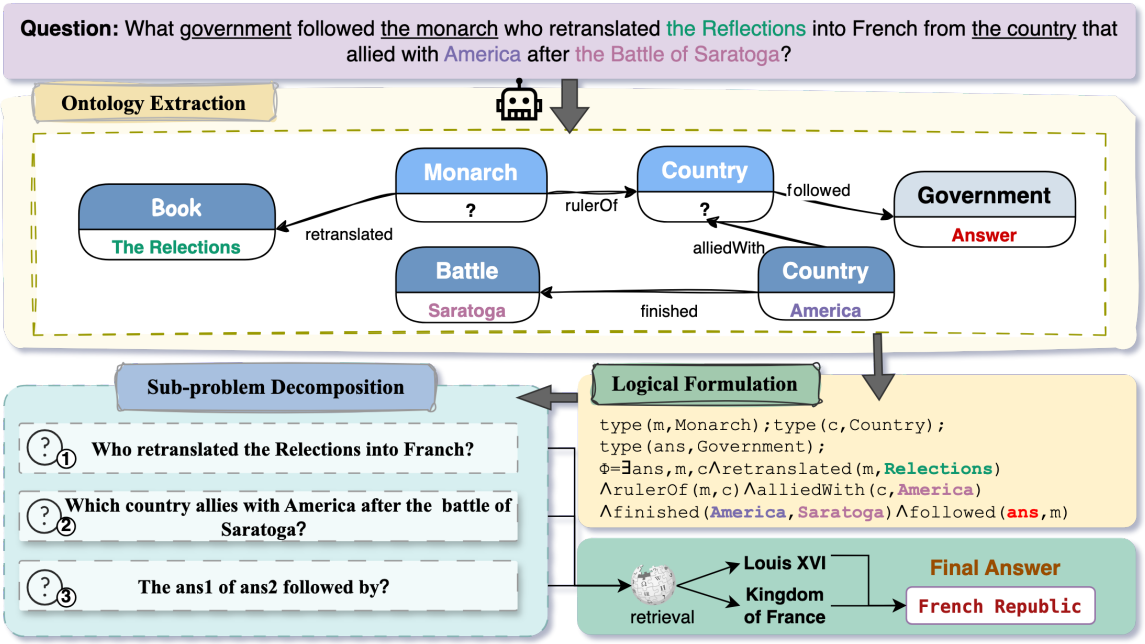

Figure 1: Overview of Our Proposed ORACLE Framework, Which Consists of Three Modules: (1) Dynamic Construction of Question-Specific Knowledge Ontologies Using LLMs, (2) Transformation of These Ontologies into FOL Reasoning Chains, and (3) Systematic Decomposition of the Original Query into Logically Coherent Sub-Questions.

- **Knowledge Graph (KG, G)**: A structured representation of factual knowledge, formally defined as a directed graph $G \subseteq E \times R \times E$.
- **Ontology** (O): A formal specification that defines the schema of a KG, including classes, relations, and their constraints, providing a structured scaffold for reasoning.
- **Class and Instance** (C): Classes group entities with common characteristics (e.g., Person, Location). The predicate $\text{type}(e, c)$ asserts that an entity $e \in E$ is an instance of class $c \in C$.

4 Methodology

Our framework for MQA (Figure 1) has three stages: *Ontology Extraction*, *FOL Construction*, and *Sub-question Decomposition*. First, an LLM extracts a question-specific ontology from the query. This ontology is then used to construct a FOL formula. Finally, the FOL formula decomposes the complex query into simpler sub-questions, which are executed to derive the final answer. Throughout this section, we use the operator $\oplus$ to denote prompt concatenation.

4.1 Ontology Extraction

Why Ontology is Necessary Direct decoding often relies on token-level similarity, leading to errors such as aligning *The Blonde from Singapore* with *The Blonde from Peking* (Appendix A.7), a phenomenon we call *logicalized semantic drift*. Introducing a lightweight ontology mitigates this issue by enforcing type constraints, relational consistency, and path stability, thereby reducing semantic drift and providing a global structural scaffold for FOL construction.

It is important to distinguish *logicalized semantic drift* from two related but fundamentally different failure modes. *Hallucination* refers to the model generating facts that do not exist—a failure at the knowledge level that occurs before reasoning begins. *Entity confusion* refers to incorrectly equating distinct entities during identification—a failure in entity resolution that precedes the reasoning chain. In contrast, logicalized semantic drift occurs at the level of reasoning path selection: the model possesses correct knowledge and correctly identifies all entities, yet during multi-hop traversal, each individual hop is subtly biased by training data distribution, causing the reasoning trajectory to gradually deviate from the question's true intent. The result is a path that appears locally coherent at each step but globally leads to an incorrect answer. This distinction motivates our ontology construc-

tion step: by explicitly anchoring the reasoning process to the semantic space of the question prior to decomposition, we suppress drift at the source rather than correcting knowledge errors after the fact. "'

Ontology Extraction The initial stage, Ontology Extraction, aims to construct a lightweight, question-specific ontology, denoted as O_Q, from the natural language question Q. This process is driven by an LLM that functions as a knowledge extractor. Unlike traditional methods that focus solely on identifying entities (E) and relations (R), our approach emphasizes the extraction of the underlying entity classes ($\mathcal{C}$). This class-centric ontology, O_Q, serves as a critical semantic and structural guide for subsequent reasoning steps. To further improve the model's planning, we also instruct the LLM to predict the class of the final answer. This extraction process is formally represented as:

$$O_Q = (C_Q, R_Q) = f_{\text{LLM}}(Q)$$

Here, $C_Q \subseteq \mathcal{C}$ is the set of relevant classes and their corresponding entities, and $R_Q \subseteq R$ is the set of relations that guides the decomposition process.

For instance, as illustrated in Figure 1, given the question Q = *"What government followed the monarch who retranslated the Reflections into French from the country that allied with America after the Battle of Saratoga?"*, this stage extracts:

► **Classes and Entities (C_Q):**

type(The Reflections, $Book$),
type(m, $Monarch$),
type(c, $Country$),
type(Saratoga, $Battle$),
type(ans, $Government$)

where m, c, and ans are variables. Notably, ans denotes the final answer entity (the Government).

► **Relations (R_Q):**

Monarch : $m \xrightarrow{\texttt{retranslated}}$ *Book* : The Reflections
Monarch : $m \xrightarrow{\texttt{rulerOf}}$ *Country* : c;
Country : $c \xrightarrow{\texttt{followed}}$ *Government* : ans;
Country : $c \xrightarrow{\texttt{alliedWith}}$ *Country:America*;
Battle:Saratoga $\xrightarrow{\texttt{finished}}$ *Country:America*.

This structure corresponds to the ontology graph shown in Figure 1, where nodes denote classes and instantiated entities, and directed edges denote relations among them.

4.2 FOL Construction

In the FOL Construction stage, the framework converts the extracted ontology O_Q into a precise and interpretable FOL formula, denoted as Φ. This transformation methodically maps the ontology's components into a logical structure: relations $r \in R_Q$ become predicates, while entity classes $c \in C_Q$ enforce type constraints on the variables. The output is a formal logical expression of the original question Q, complete with existential quantifiers ($\exists$) and a designated answer variable.

As illustrated in Figure 1, the question Q is transformed into the following formula Φ, accompanied by type declarations. The predicate names in the example are aligned with the ontology graph for clarity.

► **Type Constraints:**

type(m, $Monarch$),
type(c, $Country$),
type(ans, $Government$)

► **FOL Formula:**

$\Phi = \exists\, ans, m, c$
$\wedge$ retranslated(m, The Reflections, $French$)
$\wedge$ rulerOf(m, c)
$\wedge$ alliedWith(c, $America$)
$\wedge$ finished($Saratoga$, $America$)
$\wedge$ followed(c, ans)

For instance, the clause $rulerOf(m, c)$ uses the predicate `rulerOf` to link the variables m and c, which are constrained by the types `Monarch` and `Country`, respectively. This demonstrates how the logical formula directly mirrors the structure of the extracted ontology, thereby aiding in the decomposition of the question.

4.3 Sub-question Decomposition

The final stage, Sub-question Decomposition, breaks down the complex query into an ordered sequence of simpler sub-questions $\{Q_1, Q_2, \ldots, Q_n\}$. To achieve this, we leverage the instruction-following capabilities of an LLM. The model is provided with the original question Q, the extracted ontology O_Q, and the generated FOL formula Φ. Guided by this rich context, particularly the logical structure of Φ, the LLM

Algorithm 1 Our proposed ORACLE framework

Input: Input question Q, Large Language Model $\mathcal{LLM}$, Retriever $\mathcal{R}$, Ontology Prompt $\mathcal{P}_{Onto}$, FOL Prompt $\mathcal{P}_{FOL}$, Decomposition Prompt $\mathcal{P}_{Decomp}$
Output: Final answer, $\hat{y}$

$O_Q \leftarrow \mathcal{LLM}([\mathcal{P}_{Onto} \oplus Q])$
$\Phi \leftarrow \mathcal{LLM}([\mathcal{P}_{FOL} \oplus O_Q])$
$\{Q_1, \ldots, Q_n\} \leftarrow \mathcal{LLM}([\mathcal{P}_{Decomp} \oplus Q \oplus O_Q \oplus \Phi])$
$A_{ans} \leftarrow []$
for $i = 1$ **to** n **do**
    $Q'_i \leftarrow substitute(Q_i, A_{ans})$
    $ans_i \leftarrow \mathcal{LLM}([\mathcal{R}(Q'_i) \oplus Q'_i])$
    Append ans_i to A_{ans}
end for
$\hat{y} \leftarrow A_{ans}[n]$ ▷ Return the final sub-answer
return $\hat{y}$

formulates a multi-step reasoning plan. Each step in this plan becomes a distinct natural language sub-question, Q_i. A key feature is the use of placeholders to dynamically insert answers from preceding steps into subsequent ones, forming a coherent reasoning chain. This decomposition is formally represented as:

$$\{Q_1, Q_2, \ldots, Q_n\} = g_{\text{LLM}}(Q, O_Q, \Phi)$$

These sub-questions are then executed sequentially to derive the final answer. For each step i, the sub-question Q_i is first reformulated into Q'_i by incorporating prior answers $A_{<i} = \{A_1, \ldots, A_{i-1}\}$. A retriever, $\mathcal{R}$, fetches relevant context based on this prompt, where $C_i = \mathcal{R}(Q'_i)$. Finally, an LLM generates the answer for the current step, $A_i = \text{LLM}(Q'_i, C_i)$. This continues until the final sub-question, Q_n, is solved, returning its answer A_n as the final answer $\hat{y}$ to the original query. The overall algorithm of our proposed ORACLE framework is shown in Algorithm 1.

5 Experiments

In this section, we evaluate the performance of our method on three datasets. Furthermore, we analyze the characteristics of MQA problems and conduct detailed analysis to demonstrate the effectiveness of our approach.

5.1 Experimental Setup

5.1.1 Datasets.

We conducted experiments on the following three MQA datasets: HotPotQA (Yang et al., 2018), 2WikiMQA (Ho et al., 2020a), and MuSiQue (Trivedi et al., 2022). Similar to the previous methods (Wang et al., 2024; Shao et al., 2023b), we sampled 500 questions from the development set of each dataset. More Datasets details are provided in the Appendix.

5.1.2 Baselines.

We compare our framework to various baselines: **NoCoT**: The LLM is instructed to directly answer the input question without additional reasoning. **CoT**: Following Chain-of-Thought (Wei et al., 2022), we instruct the LLM to "Think step by step" before providing the final answer. **RAG**: The prompt sent to the LLM includes both the original question and retrieved information related to it. **ReAct**: The ReAct approach (Yao et al., 2023) guides an LLM to solve problems by cyclically generating CoTs along with an action using external tools. The results of prior cycles are utilized in the next cycle. **LPKG**: This method creates code-formatted planning demonstrations by verbalizing logical patterns from a KG (Wang et al., 2024).

5.1.3 Implementation Details.

Unless specified otherwise, we utilized *gpt-3.5-turbo*[1] as the backbone LLM. For non-retrieval baselines, we enforced standardized output formatting, including for the *DeepSeek-R1*[2] variant. All retrieval-based methods (ReAct, LPKG, and ORACLE) shared a unified retriever module. Specifically, both ReAct and LPKG employed *gpt-3.5-turbo*, with LPKG serving as a robust in-context learning baseline. Code and additional materials are available at `https://anonymous.4open.science/r/ORACLE-2D34`.

5.1.4 Evaluation Metrics

We used Exact Match (EM) and F1 Score as evaluation metrics across all MQA datasets. Both metrics first apply a normalization step that ensures a fair and case-insensitive comparison.

5.2 Experiment Results

The results in Table 1 show that ORACLE achieves state-of-the-art or highly competitive performance across HotPotQA, 2WikiMQA, and MuSiQue, with the highest average EM score of 0.340. On HotPotQA, ORACLE attains the best EM (0.396),

[1]The specific model version we use is gpt-3.5-turbo-0125.
[2]The specific model version we use is DeepSeek-R1-0528.

Method	Model	Planning	HotPotQA		2WikiMQA		MuSiQue		Average	
			EM	F1	EM	F1	EM	F1	EM	F1
NoCoT	gpt-3.5-turbo	✘	0.306	0.429	0.271	0.316	0.058	0.162	0.212	0.302
CoT	gpt-3.5-turbo	✘	0.222	0.336	0.168	0.262	0.052	0.134	0.147	0.244
RAG	gpt-3.5-turbo	✘	0.383	**0.521**	0.369	0.448	0.133	0.237	0.295	0.402
ReAct	gpt-3.5-turbo	✔	0.317	0.411	0.312	0.387	0.136	0.220	0.255	0.339
LPKG	gpt-3.5-turbo	✔	0.364	0.510	0.379	0.452	0.142	0.236	0.295	0.399
NoCoT	DeepSeek-R1	✘	0.384	0.515	0.442	0.534	0.143	0.267	0.323	0.439
ORACLE	gpt-3.5-turbo	✔	**0.396**	0.518	**0.468**	**0.547**	**0.156**	**0.242**	**0.340**	**0.436**

Table 1: Main results on multi-hop QA benchmarks. Our proposed method, ORACLE, is compared against baselines on HotPotQA, 2WikiMQA, and Musique using GPT-3.5-Turbo. The "Planning" column indicates whether a method explicitly plans its reasoning steps (✔) or not (✘). **Bold** values mark the best performance, excluding the reference row (*). The DeepSeek-R1 row, shown in gray text, is for contextual reference only and is not part of the primary comparison.

while RAG achieves a slightly higher F1; this is expected since many HotPotQA questions are pseudo multi-hop and can often be answered correctly once ample context is provided. The advantage of ORACLE is most evident on 2WikiMQA, where it sets a new state-of-the-art (EM 0.468 / F1 0.547), and it also secures the top EM score on the more challenging MuSiQue dataset.

Compared with existing approaches such as CoT, ReAct, and LPKG, our method demonstrates stronger robustness: unlike these baselines, which suffer from either lack of knowledge grounding, error propagation, or rigid planning, ORACLE consistently achieves the highest EM score across datasets thanks to its ontology-driven planning and decomposition strategy.

Dataset	**Method**	**EM**	**F1**
HotPotQA	w/o Ontology	0.338	0.424
	w/o FOL	0.304	0.408
	ORACLE	**0.396**	**0.518**
2WikiMQA	w/o Ontology	0.434	0.453
	w/o FOL	0.446	0.472
	ORACLE	**0.468**	**0.547**
MuSiQue	w/o Ontology	0.115	0.217
	w/o FOL	0.108	0.213
	ORACLE	**0.156**	**0.242**

Table 2: Ablation study results across HotPotQA, 2WikiMQA, and MuSiQue datasets.

6 Analysis

In this section, we conduct experiments to analyze the contributions of different components in our approach and further explore the factors influencing complex reasoning problems involving commonsense.

6.1 What is the contribution of each component?

To ascertain the contribution of our method's core components, we conduct an ablation study on the HotPotQA dataset. We individually remove two key modules: (1) Entity and Relation Extraction (corresponding to `w/o Ontology`) and (2) Logical Analysis (corresponding to `w/o FOL`). Table 2 shows the results, where ORACLE represents our full model.

The ablation study confirms the critical synergy between logical structure and ontological grounding. Removing the **Logical Analysis Module (w/o FOL)** leads to the most pronounced performance degradation on complex multi-hop datasets such as HotPotQA and MuSiQue. This result underscores the necessity of structured planning for navigating deep reasoning chains. Meanwhile, ablating the **Entity and Relation Extraction Module (w/o Ontology)** universally impairs results and proves even more detrimental than the removal of FOL on 2WikiMQA. These findings highlight that explicit entity anchoring is indispensable for mitigating hallucinations, particularly in entity-centric reasoning tasks.

6.2 Fine-grained Comparison

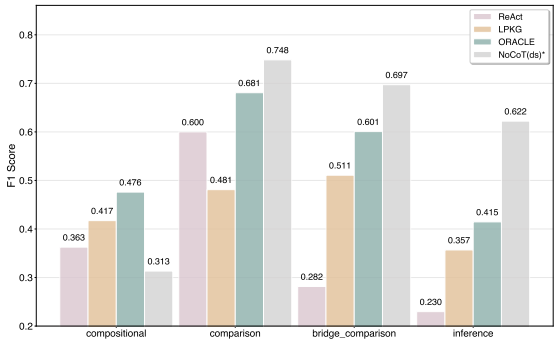

Figure 2: Fine-Grained Performance Analysis on 2WikiMQA by Reasoning Type. The Metric Shown Is F1 Score. NoCoT (ds)* Is Included for Reference Only, Where "ds" Denotes DeepSeek-R1.

To pinpoint the specific advantages of our method, we conduct a fine-grained performance analysis on the 2WikiMQA dataset, categorizing questions into four types. The results are shown in Figure 2.

Across all categories, our method, ORACLE, consistently outperforms the other planning-based methods, ReAct and LPKG. The advantage is observed on *Compositional* questions, where ORACLE achieves the highest F1 score (0.476), underscoring the effectiveness of its planning and decomposition strategy. For more complex reasoning tasks such as *Comparison*, *Bridge-Comparison*, and *Inference*, ORACLE maintains a strong lead over other planning methods. While the NoCoT (ds) reference indicates the upper-bound performance on MQA tasks, our method's consistent top-ranking performance among planning-based approaches highlights its robustness and superior reasoning capabilities across a diverse set of challenges.

6.3 Impact of Reasoning Path Quality on Performance

To quantitatively assess the quality of the generated reasoning process[3], we define a **Reasoning F_1 score** (see Appendix for implementation details). This metric measures lexical overlap between the model-generated reasoning path and the ground-truth evidence chain. A higher score indicates reasoning more consistent with the gold standard. We further segment results into high-quality ($F_1 > 0.5$) and low-quality ($F_1 \leq 0.5$) cases.

[3]Implementation details can be found in Appendix A.3.

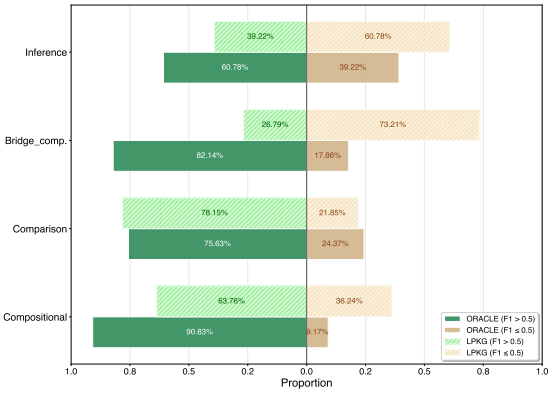

Figure 3: F1 Score Distribution for LPKG and ORACLE Methods. The Bars Illustrate the Percentage of Questions for Which Each Method Attained a High ($F_1 > 0.5$) or Low ($F_1 \leq 0.5$) F1 Score.

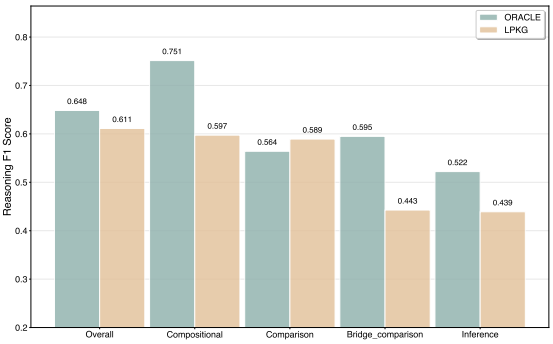

Figure 4: Comparison of Reasoning F1 Scores Between LPKG and ORACLE.

Our analysis reveals two critical findings. First, **ORACLE consistently produces higher-quality reasoning paths**. As shown in Figure 4, ORACLE achieves a higher average Reasoning F_1 than LPKG (0.648 vs. 0.611), with a notably larger proportion of high-quality paths across all categories (Figure 3). For example, on *Compositional* questions, 90.83% of ORACLE's paths exceed the 0.5 threshold, compared to only 63.76% for LPKG.

Second, **ORACLE exhibits stronger faithfulness to its reasoning path**, which is essential for reliable and interpretable AI. As illustrated in Figure 5, ORACLE's final answers directly benefit from high-quality reasoning, whereas LPKG often achieves higher answer accuracy despite flawed reasoning. For instance, on *Inference* questions, LPKG performs better with low-quality paths (0.615) than with high-quality ones (0.231), indicating that it frequently disregards its own plan and relies on parametric memory.

These findings underscore a key advantage of our approach: Ontology-guided planning enforces

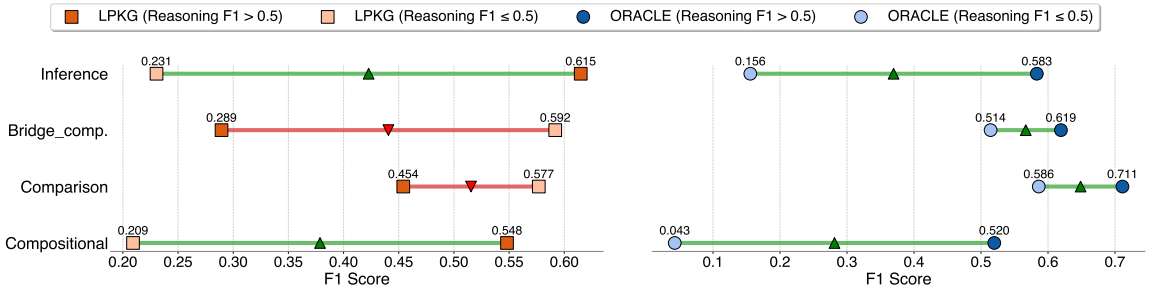

Figure 5: Impact of Intermediate Reasoning Quality. The Analysis Is Presented for Both the LPKG and ORACLE Methods Across the Four Reasoning Categories. The Line Color Indicates the Correlation Between Reasoning Quality and Final Answer Quality for That Category: Green Indicates a Positive Correlation, While Red Indicates a Weaker or Negative Correlation.

the explicit modeling of entity attributes and relations, thereby producing reasoning paths that are both higher in quality and more faithfully executed during answer generation.

6.4 Cross-Model Evaluation

To further evaluate the generalized effectiveness of our approach, we benchmarked LPKG and ORACLE on the 2WikiMQA dataset across a diverse spectrum of models, including the Qwen2.5 series, GPT-4o-mini, and Gemini-1.5-Flash. As summarized in Table 3, we report Exact Match (EM), F1, and Reasoning F1 scores, while deliberately excluding reasoning-specialized models (e.g., GPT-o1-mini) due to their frequent inability to maintain strict format adherence without task-specific fine-tuning. Our analysis reveals three primary trends: first, ORACLE consistently outperforms LPKG across most models, confirming that for sufficiently capable LLMs, ontology-guided reasoning provides a more robust structural framework for answer derivation than static knowledge injection. A notable exception occurs with Qwen2.5-7B, where LPKG leads; we hypothesize that limited parameter capacity restricts the model's ability to utilize abstract ontological guidance, whereas LPKG's explicit prompts allow smaller models to bypass complex internal reasoning via detailed surface-level cues. Finally, while performance generally correlates with model scale, Qwen2.5-14B significantly surpasses GPT-3.5, suggesting that contemporary architectural advancements can yield superior reasoning capabilities that transcend raw parameter counts.

Model	Method	EM	F1	R-F1
Closed-source Models				
GPT-3.5	LPKG	0.379	0.452	0.611
	ORACLE	**0.469**	**0.548**	**0.650**
GPT-4o-mini	LPKG	0.522	0.607	0.630
	ORACLE	**0.550**	**0.623**	**0.660**
Gemini-2.5-Flash	LPKG	0.592	**0.693**	0.705
	ORACLE	**0.612**	0.688	**0.729**
Open-source Models				
Qwen2.5-7B	LPKG	0.432	**0.533**	**0.620**
	ORACLE	**0.447**	0.507	0.598*
Qwen2.5-14B	LPKG	0.471	0.571	0.637
	ORACLE	**0.494**	**0.587**	**0.654**
Qwen2.5-32B	LPKG	0.440	0.531	0.665
	ORACLE	**0.527**	**0.635**	**0.679**
Qwen2.5-72B	LPKG	0.485	0.584	0.659
	ORACLE	**0.557**	**0.638**	**0.688**

Table 3: Comparison across models, partitioned by model availability. Note that Gemini-2.5-Flash is evaluated in non-thinking mode. The superscript $\star$ marks where LPKG outperforms ORACLE in R-F1.

6.5 Error Analysis

To further examine the limitations of our framework, we manually inspected erroneous predictions and found two recurring sources. First, the model struggles with fine-grained relational semantics: composite relations (e.g., *mother-in-law*) are sometimes reduced to direct kinship (e.g., *mother*), which derails the reasoning chain at the outset despite ontology cues. Second, numerical and temporal reasoning remains brittle: even when factual spans are correctly extracted, the system may fail to normalize and compare them, leading to inverted decisions. These observations suggest that, while ontology guidance curbs semantic drift, current LLMs still face difficulty with subtle relation understanding and numeric normalization.

Metric	ORACLE	LPKG	ReAct
Avg. Prompt Tokens (excl. context)	1,877.6	3,979.6	1,580.5
Avg. Predicted Tokens	253.8	276.7	209.6
Avg. Total Tokens	2,131.4	4,256.3	1,790.1
EM / F1	**0.468 / 0.547**	0.379 / 0.452	0.312 / 0.387

Table 4: Token consumption comparison across ORACLE, LPKG, and ReAct (retrieval context excluded).

6.6 Efficiency Analysis

To assess the computational overhead of ORACLE's three-stage pipeline, we compare token consumption across ORACLE, LPKG, and ReAct. To control for variability introduced by retrieved context length, we report instruction and reasoning token counts separately from retrieval context, enabling a cleaner comparison of inherent reasoning overhead. As shown in Table 4, ORACLE consumes significantly fewer tokens than LPKG (approximately half), owing to its more compact prompt design with fewer demonstrations. Compared to ReAct, ORACLE incurs only a modest overhead while delivering substantially better performance (+15.6% EM, +16.0% F1), demonstrating a favorable efficiency–performance trade-off.

7 Conclusion

This paper presents the ORACLE framework, which enhances LLMs on multi-hop question answering (MQA) tasks by integrating ontological guidance into the reasoning process. While LLMs excel at generating fluent answers, their greedy decoding nature often overlooks the conceptual properties of entities and the relations connecting them, leading to reasoning drift. ORACLE addresses this limitation by first constructing lightweight ontologies that explicitly encode classes, entities, and relations. These ontologies are then transformed into FOL formulas, which serve as structured scaffolds for decomposing the original question into coherent sub-questions. Experiments on standard MQA benchmarks demonstrate that this ontology-driven design not only improves performance but also yields more stable and interpretable reasoning paths, offering a promising direction for factual reasoning in LLMs.

Limitations

Our work has several limitations. Due to computational constraints, we relied on gold contexts provided by the datasets rather than full-scale Wikipedia retrieval. While this controlled setting effectively isolates reasoning logic for granular analysis, our conclusions should be interpreted within the scope of a closed-domain evaluation protocol; extending ORACLE to open-domain retrieval settings remains an important direction for future work.

Regarding comparisons with retrieval-centric systems (e.g., GraphRAG), ORACLE prioritizes query-side logical planning over data-side structural indexing. Because our method is fundamentally orthogonal to these retrieval-heavy approaches, we did not conduct extensive end-to-end benchmarking; we do, however, provide supplementary experiments in the appendix to offer partial empirical context for this comparison.

On ontology construction, our approach derives ontologies from the LLM's parametric knowledge without external verification, which may limit performance in highly specialized domains. Direct validation of generated ontologies and FOL representations through human annotation remains an open challenge. In addition, explicit mid-chain error detection and recovery are not yet incorporated; designing correction mechanisms that can handle ontology extraction failures without introducing new errors is a direction we leave to future work. Finally, while we report token consumption as a proxy for computational cost, a comprehensive efficiency analysis covering end-to-end latency and API call counts is beyond the scope of this work.

Acknowledgments

This work was supported by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant No. 2023-JC-YB-529.

References

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, and 1 others. 2023. Rest meets react: Self-

improvement for multi-step reasoning llm agent. *arXiv preprint arXiv:2312.10003*.

Jiaxin Bai, Zihao Wang, Yukun Zhou, Hang Yin, Weizhi Fei, Qi Hu, Zheye Deng, Jiayang Cheng, Tianshi Zheng, Hong Ting Tsang, and 1 others. 2025. Top ten challenges towards agentic neural graph databases. *arXiv preprint arXiv:2501.14224*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8823–8838, Online. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020a. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020b. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Tianjie Ju, Yijin Chen, Xinwei Yuan, Zhuosheng Zhang, Wei Du, Yubin Zheng, and Gongshen Liu. 2024. Investigating multi-hop factual shortcuts in knowledge editing of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8987–9001, Bangkok, Thailand. Association for Computational Linguistics.

Xinmeng Li, Mamoun Alazab, Qian Li, Keping Yu, and Quanjun Yin. 2022. Question-aware memory network for multi-hop question answering in human–robot interaction. *Complex & Intelligent Systems*, 8(2):851–861.

Zhiqiang Liu, Chengtao Gan, Junjie Wang, Yichi Zhang, Zhongpu Bo, Mengshu Sun, Huajun Chen, and Wen Zhang. 2025. OntoTune: Ontology-driven self-training for aligning large language models.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024a. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. 2024b. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2024. Enhancing reasoning capabilities of llms via principled synthetic logic corpus. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

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

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023a. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *EMNLP (Findings)*, pages 9248–9274. Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023b. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. 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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval

with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037.

Junjie Wang, Mingyang Chen, Binbin Hu, Dan Yang, Ziqi Liu, Yue Shen, Peng Wei, Zhiqiang Zhang, Jinjie Gu, Jun Zhou, Jeff Z. Pan, Wen Zhang, and Huajun Chen. 2024. Learning to plan for retrieval-augmented large language models from knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7813–7835, Miami, Florida, USA. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Juncheng Wu, Wenlong Deng, Xingxuan Li, Sheng Liu, Taomian Mi, Yifan Peng, Ziyang Xu, Yi Liu, Hyunjin Cho, Chang-In Choi, Yihan Cao, Hui Ren, Xiang Li, Xiaoxiao Li, and Yuyin Zhou. 2025. MedReason: Eliciting factual medical reasoning steps in LLMs via knowledge graphs.

Yilin Xiao, Junnan Dong, Chuang Zhou, Su Dong, Qian wen Zhang, Di Yin, Xing Sun, and Xiao Huang. 2025. Graphrag-bench: Challenging domain-specific reasoning for evaluating graph retrieval-augmented generation. *Preprint*, arXiv:2506.02404.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13326–13365, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jiacheng Ye, Chengzu Li, Lingpeng Kong, and Tao Yu. 2023. Generating data for symbolic language with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8418–8443, Singapore. Association for Computational Linguistics.

Ningning Zhang, Chi Zhang, Zhizhong Tan, Xingxing Yang, Weiping Deng, and Wenyong Wang. 2025. Credible plan-driven rag method for multi-hop question answering. *arXiv preprint arXiv:2504.16787*.

Yanan Zhang, Li Jin, Xiaoyu Li, and Honqi Wang. 2022. Edge-aware graph neural network for multi-hop path reasoning over knowledge base. *Computational Intelligence and Neuroscience*, 2022(1):4734179.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. Efficientrag: Efficient retriever for multi-hop question answering. In *EMNLP*.

A Appendix

A.1 Prompt Content

In this section, we provide the detailed prompts utilized for the different reasoning decomposition methods in our experiments.

ReAct Method Prompt

► **Input:**
Solve a question answering task with interleaving Thought, Action, Observation steps. Thought can reason about the current situation, and Action can be three types:
(1) `Search[entity]`, which searches the exact entity on Wikipedia and returns the first paragraph if it exists. If not, it will return some similar entities to search.
(2) `Lookup[keyword]`, which returns the next sentence containing the keyword in the current passage.
(3) `Finish[answer]`, which returns the answer and finishes the task.
`{examples}`
Question: {question}

The prompt for the `ReAct` method, as illustrated in the box above, guides the model to solve problems through an iterative cycle of Thought, Action, and Observation. This approach allows the model to dynamically reason and interact with an external knowledge source.

ORACLE Method Prompt

► **Input:**
Your task is to decompose complex reasoning problems into a series of sub-questions. Please follow these steps:
1. `Problem Type Identification`: Determine the problem type (2p/3p path query, 2i/3i intersection query, or ip/pi hybrid query).
2. `Entity and Relation Extraction`: List all key entities and their relationships.
3. `Logical Formula Conversion`: Convert the problem into a formal logical expression.
4. `Logical Interpretation`: Explain the meaning of the logical expression in natural language.
5. `Sub-question Decomposition`: Break down the original problem into an ordered sequence of sub-questions, clearly labeling each sub-question's answer variable.
`{examples}`
`### Your turn! Please decompose complex reasoning problem.`
Question: {question}

The prompt for the ORACLE method, as shown in the box above, presents a more structured, five-step decomposition process. This process requires the model to first analyze the problem's structure—such as type identification, entity extraction, and logical conversion—before breaking it down into a sequence of simpler sub-questions.

LPKG Method Prompt

```
from package1 import SerpAPIWrapper
from package2 import QA_LLM
search = SerpAPIWrapper()

def Search(query: str, thought: str):
    """Search relevant information about
    query based on external Search
    Engine.
    Attributes:
        query: The question you want to
    search.
        thought: The reason why this
    query is needed.
    """
    if thought is not None:
        return search.run(query)
    else:
        return "Please give your
    thought!"

def Get_Answer(query: str, info: str):
    """Get the answer of the query based
    on the information.
    Attributes:
        query: The question you want to
    search.
        info: The information relevant
    to the query.
    """
    ### Use the QA_LLM model to get the
    answer.
    return QA_LLM(query, info)

def Compare(Original_Query: str,
    Subquestions: list, Answers: list):
    """Compare the answer of the
    sub-questions and return the final
    answer of original query.
    Attributes:
        Original_Query: The original
    question.
        Subquestions: The list of
    sub-questions.
        Answers: The list of answers of
    the sub-questions.
    """
    query = Original_Query
    info = ""
    for i in range(len(Subquestions)):
        info += Subquestions[i] + ' : '
    + Answers[i] + '\\n'
    return QA_LLM(query, info)

def Intersection(Answer1: str, Answer2:
    str):
    """Find the intersection of two
    answer sets.
    Attributes:
        Answer1: The first answer set.
        Answer2: The second answer set.
    """
    List1 = Answer1.split(',')
    List2 = Answer2.split(',')
```

```
    return str(set(List1) & set(List2))

def Union(Answer1: str, Answer2: str):
    """Find the union of two answer sets.
    Attributes:
        Answer1: The first answer set.
        Answer2: The second answer set.
    """
    List1 = Answer1.split(',')
    List2 = Answer2.split(',')
    return str(set(List1) | set(List2))

def Finish_The_Plan(Answer: str):
    """Call this function to finish the
     plan and return the final answer.
    Attributes:
        Answer: The final answer of the
     original question.
    """
    return Answer

{examples}

###################
# Your turn! Just complete the code
     below and do not return other things.
###################

Original_Question:{question}
```

The LPKG method decomposes complex questions into simpler sub-questions through a structured six-step process. First, it **searches** for relevant information based on the query and provided reasoning. Then, the QA_LLM model is used to **generate answers** from the retrieved information. Next, it **compares** the answers from sub-questions and consolidates them. For certain queries, it finds the **intersection** of answer sets to identify common results, and in other cases, it computes the **union** of answers to combine multiple results. Finally, the method **returns the final answer** based on all the sub-question results. This approach enables systematic problem-solving, breaking down complex tasks into manageable steps.

A.2 More Implementation Details.

A.2.1 Datasets.

Following previous studies (Shao et al., 2023b; Aksitov et al., 2023; Wang et al., 2024), we extract the first 500 entries from the development sets of **HotpotQA** and **2WikiMQA**, respectively. For **MuSiQue**, unlike LPKG (Wang et al., 2024) or the approach by Press et al. which only utilizes 2-hop questions, we selected 500 questions in a 2:2:1 ratio of 2-hop (2p), 3-hop (3p), and 4-hop (4p) instances to ensure a more comprehensive evaluation of multi-hop reasoning.

Previous research indicates that **HotpotQA** suffers from reasoning shortcuts and "pseudo-multi-hop" phenomena (Ho et al., 2020b), often allowing models to guess answers via keyword matching rather than rigorous deduction. Similarly, **MuSiQue** focuses on evaluating retrieval capabilities under distraction, prioritizing "context localization" over "logical derivation" (Trivedi et al., 2023). In contrast, **2WikiMQA** better preserves pure reasoning structures, making it the optimal testbed to demonstrate how ORACLE corrects and refines reasoning logic.

A.2.2 Experiment Details.

Unless specified otherwise, all experiments utilized the *gpt-3.5-turbo* API as the base LLM. For non-retrieval baselines (e.g., NoCoT, CoT, and RAG), models were prompted to enclose final answers in <answer> tags to standardize evaluation. The NoCoT (DeepSeek-R1) baseline adopted the same prompt structure but utilized the *DeepSeek-R1* model.

For all retrieval-based methods (ReAct, LPKG, and our proposed ORACLE), we implemented a unified retriever module. The ReAct baseline integrated this retriever with *gpt-3.5-turbo* as its core agent. Similarly, LPKG employed *gpt-3.5-turbo* to perform its in-context learning-based planning. As a method designed for powerful, general-purpose models, LPKG provides a strong point of comparison, contrasting with our training-free framework.

A.2.3 Retriever Details.

Conventional retriever implementations for MQA typically utilize the complete Wikipedia dump, which includes *psgs_w100.tsv.gz* and *wikipedia_embeddings.tar*, as the knowledge base. However, loading the *wikipedia_embeddings.tar* file requires substantial computational resources, specifically 2x 80G A100 GPUs, making this approach prohibitively expensive.

To address this, we observed that MQA datasets generally provide a context containing the necessary paragraphs to answer the question, along with some distractor paragraphs. Consequently, we propose using this provided context as the search space for the retriever. This method offers two key advantages: it filters out the vast amount of irrelevant information present in the full Wikipedia dump and significantly reduces the demand for computational resources. By treating the provided context as the retrieval corpus, we can mitigate the possibility of

Metric	Original	Rephrased	Reordered	Mean	CV	Range
EM	0.550	0.530	0.571	0.550	3.73%	0.041
F1	0.623	0.607	0.655	0.628	3.88%	0.048
Reasoning F1	0.660	0.595	0.614	0.623	5.38%	0.065

Table 5: Prompt sensitivity analysis across three prompt variants. CV values below 10% indicate low sensitivity to prompt variations.

the retriever acting as a performance bottleneck, thereby enabling a more accurate evaluation of our proposed method's effectiveness.

In our implementation, the retriever matches the query against the provided context. Let q be the input query and $C = \{p_1, p_2, \ldots, p_n\}$ be the set of paragraphs in the given context. The retriever identifies a subset of relevant paragraphs, $C_{\text{retrieved}}$, by calculating a similarity score between the query and each paragraph. A paragraph p_i is returned if its similarity score exceeds a predefined threshold τ. This process can be formally expressed as:

$$C_{\text{retrieved}} = \{p_i \in C \mid \text{sim}(q, p_i) \geq \tau\}$$

where $\text{sim}(q, p_i)$ is the similarity function. For the RAG baseline, we simplify this process by retrieving the entire context, including both golden and distractor paragraphs. For a detailed implementation, please refer to the accompanying code.

A.3 Prompt Sensitivity Analysis

To validate that ORACLE's performance gains stem from its methodological design rather than carefully engineered prompts, we conduct a sensitivity analysis using three prompt variants: the original prompt, a rephrased variant with reworded instructions, and a reordered variant with shuffled demonstrations. We report the Coefficient of Variation (CV = std/mean $\times$ 100%) as a standardized measure of sensitivity, where a CV below 10% is generally considered indicative of low sensitivity. As shown in Table 5, all CV values fall well below 10%, confirming that ORACLE's performance is robust to prompt variations.

A.4 Generalization to GraphRAG Settings

To evaluate ORACLE's effectiveness beyond standard closed-domain benchmarks, we conduct additional experiments on the Medical subset of GraphRAG-Bench (Xiao et al., 2025), selecting 50 samples each from the Complex Reasoning and Fact Retrieval categories. We note that other question types in GraphRAG-Bench, such as summarization and generation, fall outside the scope of our framework and are therefore excluded. All experiments use GPT-4o-mini as the backbone, text-embedding-3-small as the embedding model, with top-k = 5, chunk size = 1,000, and overlap = 200. Since ORACLE does not construct an explicit graph index, index-level metrics are not applicable and are excluded from our evaluation.

As shown in Table 6, on Complex Reasoning questions, ORACLE achieves substantial gains of +9.2% in ROUGE and +7.0% in Answer Correctness over Naive RAG, demonstrating the effectiveness of our logical decomposition strategy in scenarios requiring cross-document reasoning. On Fact Retrieval, where questions are simpler and single-hop in nature, ORACLE achieves comparable Answer Correctness (0.502 vs. 0.505) with a slight improvement in ROUGE (+2.4%), indicating that our method does not degrade performance on simpler question types. The lower context relevancy and evidence recall scores are an expected trade-off: ORACLE performs multiple rounds of sub-question retrieval driven by logical decomposition, which broadens retrieved document coverage to support multi-hop reasoning at the cost of retrieval precision.

A.5 Reasoning Path and Subquestion Count Implementation

Given the scarcity of research evaluating the internal reasoning processes of LLMs, and substantial evidence showing that models can produce correct answers via fallacious reasoning, a more granular assessment is necessary. As noted by Ju et al. (2024), when faced with a query such as, "On which continent is the home country of the 2022 FIFA World Cup winner?" an LLM might not follow the correct reasoning chain, "Winner $\rightarrow$ Argentina $\rightarrow$ South America." Instead, it may leverage a shortcut learned during training, directly linking "2022 World Cup Winner" to "South America."

Consequently, we propose an evaluation frame-

Category	Method	ROUGE	Ans. Correct.	Ctx. Relev.	Evid. Recall
Fact Retrieval	Naive RAG	0.298	0.505	0.880	0.923
	ORACLE	0.322	0.502	0.667	0.731
Complex Reasoning	Naive RAG	0.187	0.533	0.792	0.842
	ORACLE	0.279	0.603	0.625	0.667

Table 6: Results on the Medical subset of GraphRAG-Bench. ORACLE shows notable gains on Complex Reasoning while maintaining comparable performance on Fact Retrieval.

work to compare the LPKG and ORACLE methods from the perspective of their reasoning processes. Our implementation leverages the 2WikiMQA dataset, which includes a `question_decomposition` field. This field serves as our ground truth, providing a list of interconnected, single-hop sub-questions (formatted as `(head entity, relation, tail entity)` triples) that constitute the correct reasoning path. We consolidate this sequence of triples into a single string to represent the ground-truth path.

For our evaluation, we transform the decomposed sub-questions and their answers from both ORACLE and LPKG into the same concatenated string of triples. To ensure fair comparison with the gold reasoning path, we first apply a preprocessing step to the generated reasoning text, including *lowercasing*, *stopword removal*, and *lemmatization*. This yields a normalized representation of the reasoning process that contains only the essential *entities* and *relations*. We then align this representation with the ground-truth reasoning path provided in the dataset, also expressed as a concatenated sequence of triples.

Formally, let $G = \{g_1, g_2, \ldots, g_m\}$ denote the set of tokens (entities and relations) in the ground-truth reasoning path, and $P_{\text{gen}} = \{p_1, p_2, \ldots, p_n\}$ denote the tokens extracted from the model-generated reasoning path after preprocessing. We compute precision P and recall R as:

$$P = \frac{|G \cap P_{\text{gen}}|}{|P_{\text{gen}}|}, \quad R = \frac{|G \cap P_{\text{gen}}|}{|G|}$$

and define the **Reasoning F1** score as:

$$\text{ReasonF1} = \frac{2 \cdot P \cdot R}{P + R}.$$

In parallel, we also compute the *sub-question count deviation* to measure the granularity of decomposition. Let k^* denote the ground-truth number of reasoning steps and k_{gen} the number generated by the model. The deviation is defined as:

$$\Delta k = k_{\text{gen}} - k^*.$$

This enables us to analyze how over-decomposition or under-decomposition impacts task performance.

A.6 Impact of Subproblem Count on Performance

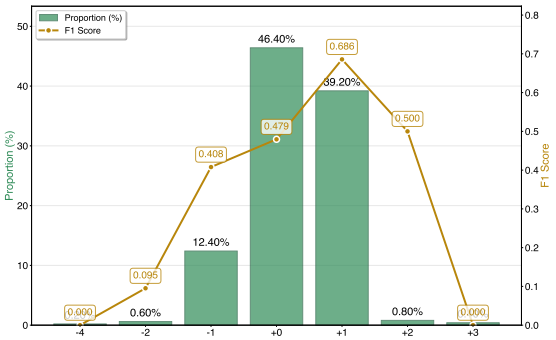

Figure 6: Impact of subproblem count deviation on F1 score (ORACLE). Bars show distribution of deviations from ground-truth counts; the line shows corresponding F1 scores.

We define *deviation* as the difference between the number of generated sub-questions and the ground-truth count (Appendix). As shown in Figure 6, 98% of cases deviate by at most one step, with perfect matches (+0) in 46.4%. Performance peaks when the model slightly over-decomposes (+1, F1=0.686), surpassing both exact matches (0.479) and under-decomposition (-1, 0.408). This indicates that including an extra step is generally safer—and often helpful—whereas missing a required step severely harms reasoning.

A.7 Case Studies

A.7.1 CoT and NoCoT

To provide a concrete illustration of our findings, we present a case study that highlights the potential drawbacks of CoT, particularly for models with more limited reasoning capabilities.

	NoCoT	CoT
Question	What nationality is Prince Napoléon Bonaparte's father?	
Prompt	You will be provided with a task description, a question and an answer format. You should only respond with the final answer in the format "Answer: ".	You will be provided with a task description, a question and an answer format. You should think step by step. In the end, put the final answer in the format "Answer: ".
Model Output	Answer: French.	The model generates a generic reasoning plan: 1. Find information about Prince Napoléon Bonaparte's father. 2. Check his father's name and background. 3. Look for his father's nationality. Answer: Italian.
Final Answer	French	Italian
Result	**Correct**	**Incorrect**

Table 7: Comparison of NoCoT and CoT

The results, detailed in Table 7, show a clear performance degradation when CoT is applied.

As shown in Table 7, the direct answering approach correctly identifies the nationality as **French**. The model effectively uses its trained knowledge to retrieve the fact directly.

In contrast, the CoT method incorrectly answers **Italian**. We hypothesize that for less capable models, CoT is counterproductive on knowledge-intensive questions because the self-generated reasoning plan flattens the final answer's logit distribution. This diffusion of attention away from the key entity and across abstract reasoning steps causes the model to retrieve a related but incorrect fact.

A.7.2 LPKG and ORACLE

To further illustrate the advantages of our ontology-driven decomposition, we present two representative case studies comparing ORACLE with the LPKG baseline.

The first case (Table 8) is a comparison-type query asking which film's director died first. LPKG mistakenly interprets the sub-question as "find another film directed by the same director," which leads to irrelevant reasoning paths. In contrast, ORACLE first extracts the ontology to anchor the key entities (films, directors, and death dates), then generates a reasoning chain that matches the ground truth. This results in a substantially higher ReasonF1, indicating closer alignment with the gold reasoning path.

The second case (Table 9) is a relation-based query about the birthplace of Marianus V of Arborea's father. Here, LPKG drifts to a descriptive path focusing on Marianus V himself, while ORACLE correctly identifies the parent entity (Brancaleone Doria) and queries his birthplace. Although minor deviations from the ground answer exist, the ontology-driven reasoning still yields a reasoning chain much closer to the ground truth, again reflected in a higher ReasonF1 score.

Together, these examples demonstrate that ORACLE not only avoids semantic drift in sub-question decomposition but also consistently produces reasoning paths that are more faithful to the intended logical structure, leading to more interpretable and reliable answers across diverse query types. Furthermore, we observe that across multiple runs, ORACLE tends to generate reasoning processes that are relatively stable, a property largely attributable to the guidance of ontology extraction at the planning stage.

	LPKG	ORACLE
Question	Which film has the director who died first, *The Piper's Price* or *The Blonde From Singapore*?	
Reasoning Process	• Which film is directed by the director of *The Piper*? → *The Faded Woman* • Which film is directed by the director of *The Blonde From Singapore*? → *The Blonde from Peking*	• Who directed *The Piper's Price*? → Joe De Grasse • Who directed *The Blonde From Singapore*? → Edward Dmytryk • When did [Joe De Grasse] die? → May 25, 1940 • When did [Edward Dmytryk] die? → July 1, 1999 • Compare death dates → May 25, 1940
Ground Truth Path	The Piper's Price → Joe De Grasse → May 25, 1940; The Blonde From Singapore → Edward Dmytryk → July 1, 1999	
ReasonF1	0.30	0.66

Table 8: Case study comparing LPKG and ORACLE on a film director comparison question.

	LPKG	ORACLE
Question	Where was the father of Marianus V of Arborea born?	
Reasoning Process	• Who is Marianus V of Arborea? → Marianus V was the Judge of Arborea • Where was the father of "Marianus V was the Judge of Arborea" born? → Castel Genovese	• Who is the father of Marianus V of Arborea? → Brancaleone Doria • Where was [Brancaleone Doria] born? → Republic of Genoa
Ground Truth Path	Marianus V of Arborea → father → Brancaleone Doria → place of birth → Sardinia	
ReasonF1	0.27	0.70

Table 9: Case study comparing LPKG and ORACLE on a familial relation query.