R2-KG: General-Purpose Dual-Agent Framework for Reliable Reasoning on Knowledge Graphs

Sumin Jo*, **Junseong Choi***, **Jiho Kim**, **Edward Choi**
KAIST
{ekrxjwh2009, quasar0311, jiho.kim, edwardchoi}@kaist.ac.kr

Abstract

Recent studies have combined Large Language Models (LLMs) with Knowledge Graphs (KGs) to enhance reasoning, improving inference accuracy without additional training while mitigating hallucination. However, existing frameworks still suffer two practical drawbacks: they must be re-tuned whenever the KG or reasoning task changes, and they depend on a single, high-capacity LLM for reliable (*i.e.,* trustworthy) reasoning. To address this, we introduce *R2-KG*, a plug-and-play, dual-agent framework that separates reasoning into two roles: an *Operator* that gathers evidence and a *Supervisor* that makes final judgments. This design is cost-efficient for LLM inference while still maintaining strong reasoning accuracy. Additionally, R2-KG employs an *Abstention mechanism*, generating answers only when sufficient evidence is collected from KG, which significantly enhances reliability. Experiments across five diverse benchmarks show that R2-KG consistently outperforms baselines in both accuracy and reliability, regardless of the inherent capability of LLMs used as the *Operator*. Further experiments reveal that the single-agent version of R2-KG, equipped with a strict self-consistency strategy, achieves significantly higher-than-baseline reliability with reduced inference cost but increased abstention rate in complex KGs. Our findings establish R2-KG as a flexible and cost-effective solution for KG-based reasoning, reducing reliance on high-capacity LLMs while ensuring trustworthy inference. [1]

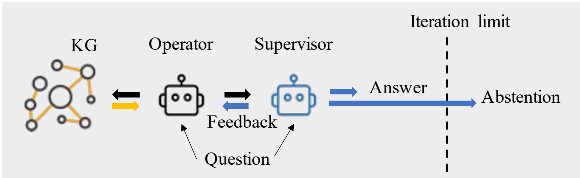

Figure 1: R2-KG: The two agents provide an 'Answer' only when they are confident enough to do so. If multiple attempts at exploration fail to gather sufficient information, it determines that it does not know and abstains from answering.

1 Introduction

Recent studies have increasingly integrated Large Language Models (LLMs) with Knowledge Graphs (KGs) to perform knowledge-grounded reasoning (Xu et al., 2024; Kim et al., 2024; Gao et al., 2024; Luo et al., 2024; Ma et al., 2024a; Wang et al., 2024). This approach maximizes reasoning performance by combining the domain-specific knowledge of KGs with the strong reasoning abilities of LLMs (Pan et al., 2024; Zhu et al., 2024).

An agent-based LLM framework treats the LLM itself as an agent that selects actions in KG and then generates the final answer (Sun et al., 2024; Ma et al., 2024b; Jiang et al., 2024). Existing agent-based LLM frameworks *claim* to be task- and KG-agnostic, yet in practice, they require non-trivial manual effort whenever either the knowledge graph changes (*e.g.*, DBpedia (Lehmann et al., 2015) → Freebase (Bollacker et al., 2008)) or new reasoning task is introduced (*e.g.*, question answering → fact verification) (Kim et al., 2023a; Sun et al., 2024; Ma et al., 2024b). For example, one of the most prominent frameworks—*Think-on-Graph* (ToG) (Sun et al., 2024) reaches its reported score only after users hand-tune exploration hyperparameters: *depth* and *width*. Also, moving to a temporal KG demands direct modification of the algorithm to inject time-aware pruning. Such hidden costs fall entirely on practitioners and undermine the promise of true generalizability.

Moreover, existing single-agent frameworks rely on one LLM to handle both KG exploration and answer generation, so their overall robustness is tightly coupled to that model's capacity. Low-

*Equal contribution.
[1]The code is available at https://github.com/ekrxjwh2009/R2-KG.git.

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 486–509
December 20-24, 2025 ©2025 Association for Computational Linguistics

capacity LLMs are more prone to early KG retrieval errors, and lack any built-in mechanism to detect or correct those mistakes. As the same agent prunes paths while exploring the KG, it cannot revisit discarded branches, resulting in an **irreversible search**. Once the forward path diverges, the system can be trapped in an incorrect subgraph without any means of recovery (Ma et al., 2024b; Tan et al., 2025; Sun et al., 2024; Jiang et al., 2024).

R2-KG eliminates these issues. We decouple evidence collection and answer validation into two collaborating agents: the *Operator* that explores the KG and logs every $<$ `entity`, `relation` $>$ decision in a persistent *chat log*, and the *Supervisor* that (i) audits the current evidence or (ii) may issue a *feedback* command to return to an earlier hop or explore an unexplored path. Through this iterative collaboration, if the framework exceeds a fixed iteration limit, it automatically abstains from answering (*i.e., Abstention mechanism*). By answering only when the evidence is sufficient—and otherwise opting not to answer—we ensure reliability. Thanks to the *Operator*'s parallel exploration strategy, **R2-KG** can keep *multiple* candidate paths alive and expand them concurrently—an ability not supported by such approaches as ToG (Sun et al., 2024) or KG-GPT (Kim et al., 2023a). Furthermore, the reasoning logic of R2-KG is frozen, thus porting R2-KG to a new KG or task requires only swapping the in-context examples in the prompt. Compared to SOTA single-agent frameworks, our work contributes:

(1) Dual-Agent Separation for Accuracy and Cost Efficiency—The low-capacity *Operator* handles KG exploration, while the high-capacity *Supervisor* provides path-level feedback and generates the final answer. The *Operator* can explore *multiple* candidate paths in parallel, while the *Supervisor* provides feedback that steers the *Operator* toward more promising branches; leveraging the *chat log*, the *Operator* can roll back to any earlier hop and re-route when necessary. Even when both agents run on low-capacity LLMs, R2-KG surpasses the best-reported performance of SOTA baselines, underscoring the strength of the architecture itself. This division increases overall accuracy while reducing the overall LLM cost.

(2) KG- and Task-Agnostic Plug-and-Play Deployment—Porting R2-KG to a new KG or reasoning task requires only swapping entity or relation names in the in-context examples, without any substantial hyperparameter tuning or algorithm edits. We evaluated on five diverse benchmarks—covering fact verification (Kim et al., 2023b), single-label QA, multi-label QA (Yih et al., 2016; Zhang et al., 2017; Talmor and Berant, 2018), and temporal QA (Saxena et al., 2021)—R2-KG surpasses strong baselines, attaining a 100% hit rate on MetaQA (Zhang et al., 2017) and up to +87.8% micro-F1 over the previous SOTA.

(3) Reliability Through the Abstention Mechanism—The *Supervisor* defers answering until evidence is sufficient; otherwise R2-KG returns **Abstain**. As a result, R2-KG offers high F1 and hit rates when it does answer, and refrains when it cannot ground a claim—maintaining user trust even when driven by low-capacity *Operator* models.

(4) Single-Agent Version with Strict Self-Consistency for Further Cost Savings—We propose an even more cost-efficient method that eliminates *Supervisor* (*i.e.*, single-agent version of R2-KG combined with strict self-consistency strategy (Wang et al., 2023b)). Here, the low-capacity *Operator* alone conducts reasoning, but ensures high reliability by requiring unanimous agreement across multiple trials before producing a result. This approach further reduces inference cost significantly, but comes with a trade-off of increased abstention rate, particularly in complex KGs with temporal information.

2 Related Works

2.1 KG-Based Reasoning with LLM

Research on KG-based reasoning tasks can be broadly categorized into three approaches: embedding-based, semantic parsing-based, and retrieval-augmented (Lan et al., 2022; Ji et al., 2024; Mavromatis and Karypis, 2024). First, the embedding-based method projects the entities and relations of a KG into an embedding space (Saxena et al., 2020). This approach effectively captures complex relationships and multi-hop connections through vector operations.

Second, the semantic parsing-based method converts the task into a symbolic logic form (*e.g.*, a SPARQL query (Pérez et al., 2009)) and executes it on the KG to derive the final answer (Sun et al., 2020; Park et al., 2021; Ye et al., 2022; Gu and Su, 2022; Yu et al., 2023). This approach has the advantage of handling complex queries, such as multi-hop reasoning, through intuitive queries that can be directly applied to the KG.

Third, the retrieval-augmented method extracts

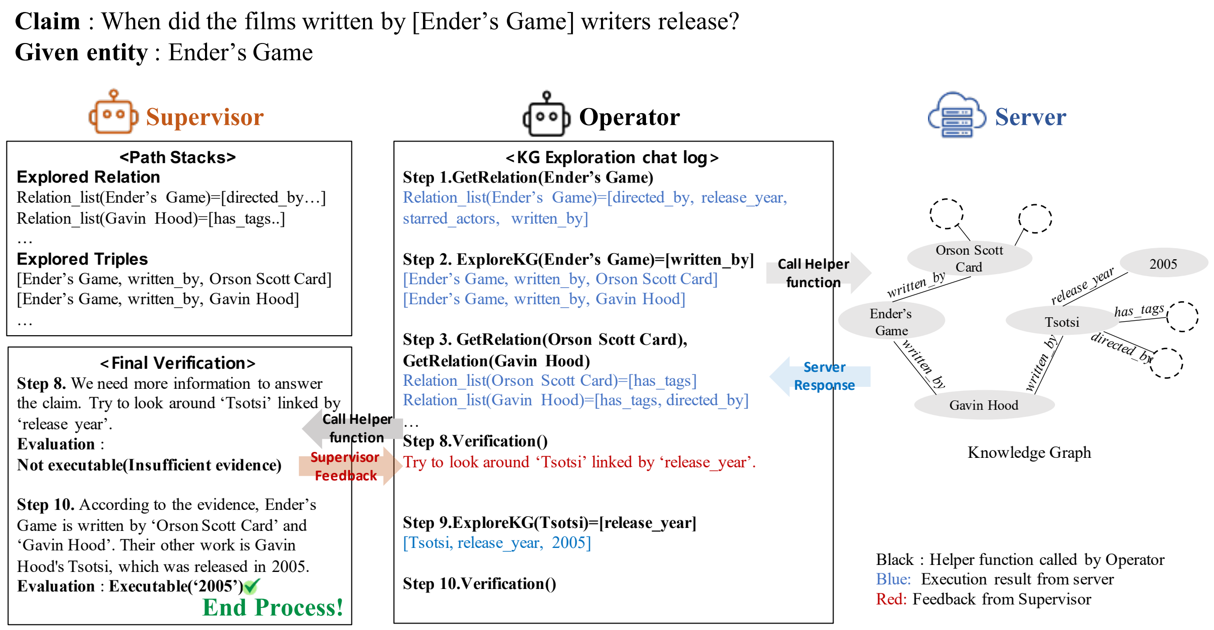

Figure 2: R2-KG solves multi-hop query through an iterative dialogue between a low-capacity *Operator* and a high-capacity *Supervisor*. The *Operator* gathers triples via *GetRelation()* and *ExploreKG()* calls, and all of the explored relations (R_k) and explored triples (G_k) are stacked in the *Supervisor*'s *Path Stacks* at every step $k < T(iteration\ limit)$. According to the *Path Stacks*, if evidence is lacking for the verification, the *Supervisor* sends feedback to the *Operator* to pursue alternative paths or roll back to an earlier hop.

relevant subgraphs from the KG to infer the answers. Recent studies explored using LLMs for both retrieval and reasoning without additional training (Kim et al., 2023a; Wang et al., 2023a; Jiang et al., 2023; Li et al., 2023; Sun et al., 2024; Ma et al., 2024b). KG-GPT (Kim et al., 2023a) proposed a three-stage framework: Sentence Segmentation, Graph Retrieval, and Inference. ToG (Sun et al., 2024) and ToG-2.0 (Ma et al., 2024b) introduced frameworks that conduct reasoning by pruning relations and entities during KG exploration. While these LLM-based methods enhance the performance of KG-based reasoning, they struggle to adapt to new KG structures or tasks. Also, these frameworks can explore the KG only up to the fixed hyperparameters (*e.g.,* depth, width, top-k), and because they do not retain the full history of visited triples, they cannot return to earlier paths. As a result, potentially relevant branches can be missed. To overcome these limitations, we introduce R2-KG, a truly generalizable framework that enables more accurate and efficient KG exploration.

2.2 Enhancing Model Reliability via Abstention Mechanism

To mitigate LLM hallucination, the *abstention mechanism* has been adopted as a strategy to enhance reliability (Wen et al., 2024b). This mechanism allows the model to refrain from answering when the input query is ambiguous (Asai and Choi, 2021; Cole et al., 2023), goes against human values (Kirk et al., 2023), or exceeds the model's knowledge scope (Feng et al., 2024). The *abstention mechanism* has been actively explored in LLM-based question-answering tasks, particularly for long-document processing QA (Buchmann et al., 2024) and uncertainty estimation (Amayuelas et al., 2024; Wen et al., 2024a; Yang et al., 2024; Tomani et al., 2024), demonstrating notable improvements in reliability. However, its application in KG-based reasoning remains largely unexplored. We introduce *Reliable KG-Based Reasoning Task*, the first approach to integrate the *abstention mechanism* into KG-based reasoning.

3 Reliable KG-Based Reasoning Task

3.1 Task Definition

In this study, we propose the *Reliable KG-Based Reasoning Task* for the first time. This task serves as a benchmark for measuring reliability in KG-based reasoning, particularly in domains where trustworthy answers are critical, such as industrial applications and fact verification that utilize KGs. By evaluating reliability, this enables the selection of an appropriate framework based on the specific context. Unlike existing KG-based reasoning tasks that focus on generating a definitive answer *a* (*e.g.*, True / False in fact verification or a direct response in QA) for a given query *q* (*e.g.*, a query in fact verification or a question in QA), our task introduces the option to *abstain* when uncertainty arises. This allows the system to either withhold a response when sufficient evidence cannot be retrieved from the KG or avoid providing an unreliable answer based on ambiguous evidence.

3.2 Metrics

To evaluate the KG-based reasoning task incorporating the *abstention mechanism*, we measure four key metrics:

Coverage: The fraction of samples for which a final answer is generated (*i.e.*, the ratio of non-abstained samples).

$$\text{Coverage} = \frac{|\mathcal{S}|}{|N|}$$

where $\mathcal{S}$ denotes the set of non-abstained samples, and N represents the set of all samples, including abstained and non-abstained cases.

Micro F1 Score: Computed on $\mathcal{S}$ in multi-label tasks using TP_i, FP_i, FN_i, which represent the True Positives, False Positives, and False Negatives for each sample i, respectively.

$$\text{Micro F1} = \frac{2 \times \text{Total Precision} \times \text{Total Recall}}{\text{Total Precision} + \text{Total Recall}}$$

$$\text{Total Precision} = \frac{\sum_{i \in \mathcal{S}} TP_i}{\sum_{i \in \mathcal{S}} (TP_i + FP_i)}, \text{Total Recall} = \frac{\sum_{i \in \mathcal{S}} TP_i}{\sum_{i \in \mathcal{S}} (TP_i + FN_i)}$$

Samplewise F1 Score: Calculated on $\mathcal{S}$ in multi-label tasks by computing F1 score for each sample and averaging over $\mathcal{S}$.

$$\text{Samplewise F1} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}, \text{Recall}_i = \frac{TP_i}{TP_i + FN_i}$$

Hit Rate: Applicable to both single-label and multi-label tasks. It is counted if any predicted label matches a ground-truth label. Note that the hit rate is the accuracy in binary tasks.

$$\text{Hit rate} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \mathbb{1}(\hat{y}_i \in Y_i)$$

where $\mathbb{1}(\cdot)$ is the indicator function, $\hat{y}_i$ is one of the framework's predicted label for sample i and Y_i is the set of ground truth labels for sample i.

4 Method

Our R2-KG consists of three components: An *Operator*, which explores the KG via helper functions; a *Server*, which provides requested function output; and a *Supervisor*, which offers feedback or generates the final answer. Within an iteration limit *T*, the three components iteratively interacts, gathering triples G_t or relations R_t, at each step t. The *Supervisor* outputs the final answer once sufficient evidence is collected. If no answer is produced within *T*, the system returns an *Abstention*, indicating insufficient understanding of the query.

4.1 Operator

By leveraging helper functions (described below), the system retrieves relevant subgraphs from the KG. When the *Operator* requests a function call, the *Server* responds, and their interactions are accumulated in *chat log* at each step *t* for future reference.

For multi-hop reasoning, R2-KG iteratively expands the subgraphs by accumulating relevant triples. Given a query where entity e_0 and e_n are connected through n-hops, the intermediate entities are unknown. At an arbitrary step k, the *Operator* maintains $E_{seen}^{(t=k)} = \{e_0, \ldots, e_{m-1}, e_m\}$, which is the set of entities explored up to the previous step, where $E_{seen}^{(t=0)} = \{e^0\}$. Each $e_i \in E_{seen}$ is associated with relations $R(e_i) = \{r_{i(1)}, r_{i(2)}, \ldots, r_{i(n)}\}$. In the next step, *Operator* selects a relevant $e^* \in E_{seen}$ and one or more relevant relations $R^* \subseteq R(e^*)$, retrieves the corresponding tail entities, and get a new triple set: $\{(e^*, r^*, e_{m+1}) \mid r^* \in R^*\}$. This process continues until e_{m+1} matches e_n.

By structuring reasoning in this way, R2-KG ensures that each step builds upon *chat log*, improving both exploration efficiency and reasoning accuracy. The *Operator* can, at each step-t, invoke *multiple* following helper functions in parallel enabling simultaneous exploration of several graph branches and accelerating KG search.

GetRelation(e^*): The *Server* returns all relations $R(e^*)$ connected to e^* in the KG as follows:

$$e^* = \arg\max_{e \in E_{seen}} EntScore(e, q)$$
$$R(e^*) = \{r_i \mid (e^*, r_i, e_j) \in KG, \forall e_j\}$$

The *Operator* selects e^* that is most relevant to q among E_{seen} using $EntScore(e, q)$, which is a function that evaluates the relevance between e and

q. Note that $EntScore(\cdot)$ is based not on an explicit implementation but on the inherent language understanding of the *Operator*.

ExploreKG$(e^*, R^*(e^*))$: The *Server* returns $G(e^*, R^*(e^*))$, a set of all triples such that $e^* \in E_{seen}$ is connected to a tail entity e_j via the relation $r_i \in R^*(e^*)$. Note that $R^*(e^*)$ is a subset of $R(e^*)$, which is returned by *GetRelation()* chosen by *RelScore()* as below:

$$R^*(e^*) = \{r \mid r \in R(e^*),\ RelScore(r, q) > \text{threshold}\}$$
$$G(e^*, R^*(e^*)) = \{(e^*, r_i, e_j) \mid r_i \in R^*(e^*),\ e_j \in KG\}$$

$RelScore(r, q)$ evaluates the relevance between r and q based on the inherent language understanding of the *Operator*. Along with the threshold, it is implicitly applied during the *Operator*'s linguistic reasoning process to select several relations relevant to q.

Verification(G_k, R_k): If the collected evidence is deemed sufficient, *Operator* invokes the *Supervisor*. The *Operator* provides the explored triples G_k and explored relations R_k gathered up to the current step $k(< T)$ to the *Supervisor*. If the *Supervisor* gives back an answer, the process terminates; otherwise, if feedback is given, the next iteration continues.

$$R_{\text{k}} = \bigcup_{t=1}^{k} R_t(e^*), \quad G_{\text{k}} = \bigcup_{t=1}^{k} G_t(e^*, R^*(e^*))$$

4.2 Supervisor

The *Supervisor* performs its role only when the *Operator* invokes *Verification(G_k, R_k)*. Upon invocation, the *Supervisor* receives the G_k and R_k and returns one of two possible outcomes to the *Operator*:

1) Sufficient Evidence (answer): If sufficient information is available, the *Supervisor* generates a prediction and returns it to the *Operator*. The final reasoning path[2] optimized for answer generation is constructed by the *Supervisor* based on its judgment, using G_k.

2) Insufficient Evidence: If the evidence is lacking, based on G_k, R_k, and q, the *Supervisor* suggests new or previously pruned *entity–relation* pairs, enabling the *Operator* to roll back to any earlier hop or branch into unseen entities before resuming the search[3].

[2]You can find the example of final reasoning path of *Supervisor* from Appendix E

[3]You can find the example of *Supervisor*'s feedback for *Operator* in Appendix J

4.3 Configurable Iteration Limit

During KG exploration, R2-KG requires at least two iterations to traverse a single hop—first using *GetRelation($\cdot$)*, then *ExploreKG($\cdot$)*. Therefore, if a query requires H hops, we recommend setting the iteration limit $T \geq 2H$. However, since R2-KG can issue multiple helper function calls in parallel within an iteration and flexibly reuse partial evidence, it can complete multi-hop reasoning with fewer than $2H$ steps.

Unlike prior methods where hyperparameters (*e.g.*, depth in ToG, top-k in KG-GPT) directly constrain the discoverable reasoning paths, T in R2-KG is designed as a reliability knob. A lower T favors high-precision decisions by limiting uncertain inferences, while a higher T enhances coverage through broader evidence exploration.

5 Experiments

5.1 Datasets

To demonstrate that R2-KG is a plug-and-play approach independent of task and KG variation, we use five challenging benchmarks with diverse query difficulty, KG structures, and task formats. Table 1 shows the features and statistics of the dataset we used. WebQSP (Yih et al., 2016) is a semantic parsing QA dataset, and CWQ (Talmor and Berant, 2018) builds on it with more complex questions involving compositional and comparative reasoning. MetaQA (Zhang et al., 2017) dataset has 1-hop, 2-hop, and 3-hop questions, we focus on most challenging 3-hop task[4]. CRONQUESTIONS (Saxena et al., 2021) is a temporal reasoning benchmark, we used three question types (*i.e.*, simple time, simple entity, time join), excluding others due to missing labels (details in Appendix B). FactKG (Kim et al., 2023b) contains the most structurally complex multi-hop queries among publicly released benchmarks to date with five reasoning types (*i.e.*, one-hop, conjunction, existence, multi-hop, negation). To reduce computational costs, we sample 1,000–1,500 instances from large test sets[5].

5.2 Baselines

For comparison, we set KG-GPT (Kim et al., 2023a), and ToG (Sun et al., 2024) as baselines, as both can handle various KG structures and tasks

[4]MetaQA 1-hop and 2-hop tasks are covered in Appendix A

[5]Full-dataset experiments employing GPT-4o mini for both agents are provided in Appendix I

Dataset	Feature / Base	Answer Type	Total # Test Set	Used # Test Set
WebQSP	Freebase	Entity (M)	1639	1639
CWQ	Freebase	Entity (M)	3531	1000
MetaQA 3-hop	Movie-related	Entity (M)	14274	1000
FactKG	DBpedia	Boolean	9041	1000
CRONQUESTIONS	Wikidata	Entity/Number (S, M)	16690	1450

Table 1: Dataset Statistics. (M): Multi-label QA, (S): Single-label QA.

to some extent. For fairness, all baselines are also evaluated with both low- and high-capacity LLMs, consistent with the R2-KG setup. KG-GPT is a general framework adaptable for fact verification and QA tasks. However, it does not explicitly incorporate an *abstention mechanism*, therefore we account for implicit *Abstention* when it is unable to generate an answer due to token length constraints or formatting issues. Additionally, due to the structural modifications required to adapt KG-GPT for WebQSP and CWQ, we did not conduct experiments on this dataset. ToG also employs an LLM agent for both KG exploration and answer generation. When ToG exceeds the depth limit (*i.e.*, hop limit), it relies on the LLM's parametric knowledge to generate answers, which we treat as *Abstention*. However, we could not conduct an experiment for CRONQUESTIONS because ToG cannot handle time-structured KG queries without fundamental algorithmic changes. Additionally, we assess GPT-4o-mini's ability to answer without KG access, treating predictions as correct if they convey the same meaning as the ground truth (*e.g.*, '*America'* equivalent with '*USA'*). For details on the modifications made to baselines, refer to Appendix N.

5.3 Experimental Setting

For the *Operator*, we use six LLMs. We employ GPT-4o mini and GPT-4o (OpenAI, 2024a,b) as API-based models, and LLaMA-3.1-70B-Instruct (Meta, 2024), Mistral-Small-Instruct-2409 (Mistral, 2025), Qwen2.5-32B-Instruct, and Qwen2.5-14B-Instruct (Qwen, 2025) as open-source LLMs. The maximum token length was set to $8,192$ for CRONQUESTIONS and FactKG, and $16,384$ for MetaQA, WebQSP, and CWQ. Top-p and temperature were both set to 0.95. For the *Supervisor*, we use GPT-4o. In the main experiment, T was set to 15. All experiments were conducted on a system equipped with two NVIDIA A100 GPUs and four NVIDIA RTX A6000 GPUs. Check models' spec in Appendix F.

6 Main Results

6.1 Performance of R2-KG

As shown in Table 2, R2-KG consistently outperforms baselines in F1 score and achieves higher hit rates on four out of five benchmarks. Even with low-capacity LLMs as the *Operator*, R2-KG surpasses ToG and KG-GPT, which fully rely on GPT-4o (*i.e.*, high-capacity LLM) throughout the reasoning process. Additionally, R2-KG achieves a hit rate of over 90% in three out of the five benchmarks, with MetaQA 3-hop reaching 100%. On WebQSP, ToG with GPT-4o mini marginally outperforms R2-KG in terms of hit rate, but R2-KG achieves significantly higher F1 scores, which is a more suitable metric for multi-label QA, demonstrating its superior reasoning performance. This highlights the advantage of R2-KG not only in single-label QA but also in multi-label QA. WebQSP and CWQ yielded hit rates below 90%, CRONQUESTIONS showed micro F1 under 43% across all models—a pattern further analyzed in Appendix G. The strong performance of R2-KG can be attributed to its *Operator*'s ability to accumulate and utilize information from previous hops in multi-hop reasoning. Within a given T, the framework can revisit and adjust incorrect paths from prior steps, dynamically selecting alternative paths as needed. Furthermore, during inference, the *Supervisor* is not restricted to a single reasoning path but can flexibly combine relevant triples, leading to more accurate reasoning and answer generation.

To ensure that R2-KG 's performance does not rely on the capability of high-capacity LLMs, we conducted additional experiments by varying not only the *Operator* but also the *Supervisor* across different model scales. The results show that even when the Supervisor is a low-capacity LLM, R2-KG still achieves higher F1 scores and hit rates than the baseline. Please check the Appendix C.

6.2 Coverage Across Different LLMs

Note that R2-KG's coverage is the highest across all cases when using GPT-4o as the *Operator*. When using relatively low-capacity LLMs, the coverage decreases in varying degrees. The reason why high-capacity LLMs as a *Operator* achieve higher coverage is twofold: First, they excel at collecting key evidence, allowing them to request *Verification(·)* at the optimal moment. Second, their strong language understanding enables them to effectively use the feedback provided by the *Supervi-*

Method	Utilized Model		WebQSP				CWQ				MetaQA 3-hop				CRONQUESTIONS				FactKG	
	Operator	Supervisor	Cvg	F1 (M)	F1 (S)	Hit	Cvg	F1 (M)	F1 (S)	Hit	Cvg	F1 (M)	F1 (S)	Hit	Cvg	F1 (M)	F1 (S)	Hit	Cvg	Hit
w/o KG	GPT-4o mini	–	99.0	12.5	25.8	36.9	95.3	25.8	28.8	36.3	96.2	7.0	14.5	36.6	100	4.0	15.0	24.0	100	50.0
KG-GPT	Mistral-Small	–	–	–	–	–	–	–	–	–	100	6.8	21.4	54.6	95.7	7.8	49.9	60.0	55.4	57.6
KG-GPT	GPT-4o mini	–	–	–	–	–	–	–	–	–	100	12.6	36.6	97.9	100	11.7	63.4	91.7	100	63.3
KG-GPT	GPT-4o	–	–	–	–	–	–	–	–	–	100	12.6	36.2	97.3	100	10.6	60.3	83.8	100	79.9
ToG	Mistral-Small	–	30.5	24.1	65.9	82.6	15.4	48.2	57.8	64.3	24.1	13.2	31.2	62.2	–	–	–	–	52.8	69.5
ToG	GPT-4o mini	–	53.1	21.7	72.8	**90.7**	33.2	57.1	67.7	76.5	30.5	13.6	28.5	67.2	–	–	–	–	35.8	83.5
ToG	GPT-4o	–	58.8	21.9	69.6	89.1	40.3	57.7	67.8	76.5	24.5	15.6	44.0	95.5	–	–	–	–	50.6	86.8
R2-KG	Qwen2.5-14B	GPT-4o	76.4	75.7	80.9	87.9	51.5	73.6	76.7	82.3	82.9	90.3	94.5	97.9	83.7	40.4	89.0	99.6	55.8	**93.4**
R2-KG	Qwen2.5-32B	GPT-4o	81.5	**79.4**	**83.0**	89.5	59.4	69.3	77.7	82.8	96.5	**98.3**	99.1	**100**	87.8	36.0	86.6	**99.8**	64.1	93.2
R2-KG	Mistral-Small	GPT-4o	76.3	76.7	82.3	89.4	40.3	**76.9**	**79.8**	**85.1**	75.0	94.5	96.3	99.3	65.9	33.1	87.6	99.4	43.2	93.1
R2-KG	Llama-3.1-70B	GPT-4o	81.0	78.4	80.3	87.7	62.8	75.6	79.0	84.2	94.9	97.7	98.7	99.9	84.1	**42.2**	**89.9**	99.7	57.3	92.7
R2-KG	GPT-4o mini	GPT-4o	81.3	73.6	80.1	88.4	63.1	69.6	77.6	82.4	94.6	95.7	97.6	99.9	90.4	34.3	85.6	99.4	70.2	92.5
R2-KG	GPT-4o	GPT-4o	85.3	71.1	81.4	89.1	76.2	71.2	76.7	82.3	98.3	**98.3**	**99.2**	99.9	90.8	33.6	85.3	99.5	77.8	93.1

Table 2: Performance of baselines and R2-KG on the five KG-based reasoning benchmarks. We denote the **best** and second-best method for each metric (except coverage). Cvg: Coverage, F1 (M): Micro F1 score, F1 (S): Samplewise F1 score.

sor. Table 2 shows that even with a low-capacity LLM as the *Operator*, R2-KG maintains a high F1 score and hit rate despite reduced coverage. This highlights the advantage of R2-KG's separation of the *Operator* and *Supervisor*. Since R2-KG maintains answer reliability while only affecting coverage, users can confidently choose an *Operator* based on their budget constraints.

6.3 Case analysis of Abstention

Even when T is high, reasoning may still fail, leading to abstention. The most common cases are: (1) Repeated helper function requests—The *Operator* redundantly calls the same function across multiple steps, even after retrieving the necessary information in previous steps. (2) Failure to interpret *Supervisor*'s feedback—The *Operator* struggles to incorporate the *Supervisor*'s instructions, especially when directed to collect additional information about a specific entity's relation, failing to refine exploration in later steps. (3) Failure to extract an answer despite sufficient evidence—When the retrieved triple set is overly large, the *Supervisor* may misinterpret relationships between triples, leading to incorrect judgment. (4) Incorrect function call format—The *Operator* does not follow the predefined format when calling a helper function, causing parsing issues that prevent information retrieval.

6.4 LLM Usage Comparison

Table 3 shows that *Operator* and *Supervisor* average 5.94–8.63 and 1.04–1.43 calls per q, respectively; by contrast, KG-GPT makes at least 3 high-capacity LLM calls, and ToG makes minimum 4 to maximum 25 such calls in the whole reasoning process. R2-KG employs *Low/High-Capacity LLM Separation* for accuracy and cost efficiency, significantly reducing high-capacity LLM usage to an average of 1.28 calls per q, making it both cost-effective and superior in performance.

Dataset	Operator Call	Supervisor Call
WebQSP	5.94	1.04
CWQ	8.38	1.28
MetaQA 3-hop	8.63	1.38
FactKG	8.21	1.43
CRONQUESTIONS	7.34	1.27

Table 3: Number of LLM calls per sample for *Operator* and *Supervisor* in different datasets

7 Further Analysis

7.1 Effect of Iteration Limit

Figure 3 illustrates the impact of T on coverage, F1 scores, and hit rate. At $5 \leq T \leq 15$, coverage improves, whereas F1 scores and hit rate slightly decline. Lower $T(=5)$ causes early termination, leading to lower coverage but higher accuracy on simpler queries (exception arises in CWQ—due to extremely low coverage, few errors affect overall performance). At $10 \leq T \leq 15$, increased evidence collection enhances coverage, though accuracy slightly drops as queries grow more complex. Beyond 20 iterations, coverage stabilizes while F1 scores and hit rates marginally decrease. This suggests that the optimal iteration range is 10-15 for the benchmarks we used, as further steps mainly introduce redundant exploration that is unhelpful for reasoning.

7.2 Single-Agent Version of R2-KG with Strict Self-Consistency

To further reduce the cost of using a high-capacity LLM as the *Supervisor*, we leverage a self-consistency (Wang et al., 2023b) strategy where

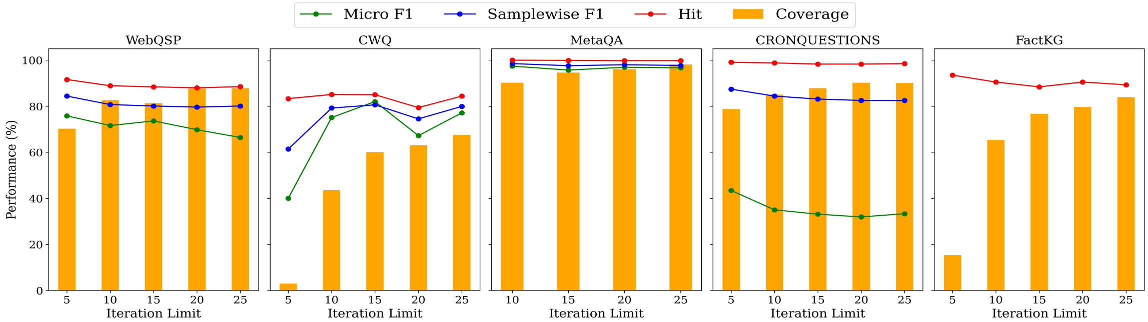

Figure 3: Changes in coverage, F1 Scores, and hit rate based on Iteration Limit

Method	Operator	Supervisor	Total
Dual-agent	9.89 s	3.17 s	13.06 s
Single-agent	12.47 s	–	12.47 s

Table 4: Time spent per method on CWQ

Method	Operator	Supervisor
Dual-agent	18,987	4,912
Single-agent	19,309	–

Table 5: Token usage per method on CWQ

the *Operator* handles both evidence collection and answer generation (*i.e.*, single-agent version of R2-KG). Without the *Supervisor*, the *Operator* assesses evidence sufficiency and generates answers within *Verification(·)*. The reasoning process runs three trials per instance with $T = 10$, following these rules; First, unlike the typical majority-based self-consistency strategy, our approach enforces a stricter unanimous agreement criterion for the final prediction. Second, if no agreement is reached or if *Abstention* appears in any attempt, the final prediction is also *Abstention*. We apply three reasoning path strategies; Multi-Prompts (distinct in-context examples for the same query), Query Paraphrasing (semantically equivalent query variations), Top-p/Temperature Variation (sampling diversity)[6].

Table 6 shows a significant decrease in coverage compared to the dual-agent version of R2-KG, while F1 scores and hit rate were comparable or slightly improved except for MetaQA 3-hop and CRONQUESTIONS. Despite this, it still significantly outperformed baselines, achieving 100% on MetaQA 3-hop and micro F1 gains (WebQSP +55.8%, CWQ +22.2%, MetaQA 3-hop +80.2%, and CRONQUESTIONS +37.5%) compared to the baselines. These results demonstrate that a single-agent variant of R2-KG can achieve higher-than-baseline answer reliability at even lower cost. Multi-Prompts generally showed strong performance across all datasets. However, relying solely on low-capacity LLMs limits adaptability to more complex KGs like CRONQUESTIONS (*i.e.,* KGs that require reasoning over temporal constraints and time-sensitive relations), and stricter filtering inevitably leads to reduced coverage and overall utility compared to the dual-agent setup.

7.3 Comparison of Time Consuming and Token Usage of Dual/Single Version of R2-KG

In the dual-agent setup, the *Operator* calls the *Verification(·)* function to request the *Supervisor* 's help in gathering sufficient evidence. In the single-agent variant, however, the *Operator* must generate an answer directly by referencing the provided triples when *Verification(·)* is invoked. We measured the average time spent and token usage on CWQ with GPT-4o mini as the *Operator* and GPT-4o as the *Supervisor*. Table 4 shows that the latency gap between the dual-agent and single-agent versions is under one second. This represents only a minor portion of the total query-processing time, supporting our claim that the dual-agent framework's latency is not practically problematic. We also compared token usage between the dual-agent and single-agent settings. As shown in Table 5, the dual-agent framework used a total of 18,987 tokens for the *Operator* and 4,912 for the *Supervisor*, while the single-agent version used 19,309 tokens in total—resulting in a difference of 4,590 tokens. Restricting the high-capacity LLM to the supervisor role in the dual-agent version markedly lowers token usage compared to the single-agent version. The advantage of the dual-agent framework is therefore evident, as it guarantees both high coverage and reliability. Nonetheless, by using low-capacity LLM as a *Operator*, the single-agent version with

[6]Detailed experimental settings and prompt examples are provided in Appendix K

Reasoning Path Strategy	Utilized Model	WebQSP				CWQ				MetaQA 3-hop				CRONQUESTIONS				FactKG	
	Operator	Cvg	F1 (M)	F1 (S)	Hit	Cvg	F1 (M)	F1 (S)	Hit	Cvg	F1 (M)	F1 (S)	Hit	Cvg	F1 (M)	F1 (S)	Hit	Cvg	Hit
Multi Prompts	Qwen2.5-14B	54.0	70.4	80.4	92.4	34.7	73.3	77.8	85.9	–	–	–	–	69.7	26.7	79.0	97.4	44.5	94.4
	Qwen2.5-32B	62.1	69.2	81.2	93.6	40.9	71.9	80.0	87.5	80.8	<u>95.7</u>	<u>96.3</u>	**100**	78.2	25.8	75.9	**98.9**	67.3	92.3
	Mistral-Small	40.5	<u>76.3</u>	<u>84.0</u>	**95.9**	21.4	66.9	75.7	85.0	58.1	86.4	88.5	**100**	39.0	<u>47.7</u>	**86.8**	98.6	25.5	93.3
	Llama-3.1-70B	57.2	74.1	81.6	93.3	46.5	**79.9**	<u>82.5</u>	87.5	86.7	93.4	94.7	**100**	73.9	30.0	<u>84.6</u>	**98.9**	58.2	93.8
	GPT-4o mini	65.1	71.9	83.3	93.8	36.0	74.3	80.0	87.5	84.1	90.9	95.2	**100**	64.3	18.0	72.5	96.8	76.1	92.1
Paraphrasing	Qwen2.5-14B	54.7	61.0	78.0	94.0	33.4	71.8	76.1	83.8	–	–	–	–	54.3	40.6	81.7	97.1	48.6	94.2
	Qwen2.5-32B	68.4	64.8	78.4	92.6	39.9	75.3	78.4	85.5	77.7	94.8	95.7	99.9	95.6	27.3	76.7	97.5	47.9	**95.3**
	Mistral-Small	50.5	70.3	81.4	93.8	19.0	63.5	72.1	82.6	60.4	82.5	84.8	99.7	58.1	47.5	84.3	98.2	24.6	95.1
	Llama-3.1-70B	57.0	75.3	82.8	94.4	44.4	79.4	81.9	87.8	89.5	**95.8**	96.1	**100**	78.8	37.5	73.1	97.7	50.2	94.8
	GPT-4o mini	69.6	60.5	81.2	92.5	44.0	69.1	80.3	**88.6**	82.4	92.9	95.8	**100**	94.1	29.4	80.1	96.7	71.7	92.5
Top-p / Temperature	Qwen2.5-14B	51.3	67.6	80.1	94.3	36.4	73.8	78.8	86.5	–	–	–	–	60.0	46.1	84.0	96.9	42.2	92.9
	Qwen2.5-32B	67.5	63.8	78.9	91.9	41.2	71.3	78.5	85.9	74.5	93.0	94.9	99.9	13.3	12.7	29.7	92.7	48.9	92.9
	Mistral-Small	51.4	75.3	**84.2**	<u>95.0</u>	14.1	59.9	71.4	83.7	61.0	87.6	89.8	99.8	71.9	38.1	84.1	98.5	15.7	90.4
	Llama-3.1-70B	63.1	**79.9**	82.5	93.1	49.1	78.6	81.0	86.4	91.0	95.2	**96.5**	**100**	94.4	**49.2**	83.2	97.2	58.6	93.3
	GPT-4o mini	69.1	61.9	79.8	91.1	42.5	<u>79.8</u>	**83.0**	<u>88.2</u>	84.4	90.4	94.1	99.2	95.9	31.2	80.1	96.8	37.9	**95.3**

Table 6: Performance of single-agent version of R2-KG with self-consistency on the five KG-based reasoning benchmarks. We denote the **best** and <u>second-best</u> method for each metric (except coverage). Cvg: Coverage, F1 (M): Micro F1 score, F1 (S): Samplewise F1 score.

self-consistency remains a practical alternative for cost-sensitive applications; although its coverage is somewhat reduced, it still ensures high reliability, making it a viable option when cost is the primary constraint.

8 Conclusion

We propose R2-KG, the first general KG-based reasoning framework with an *abstention mechanism*, ensuring the reliability for various KG-based reasoning tasks. Separation of *Operator* and *Supervisor* reduced high-capacity LLM usage, leading to a cost-effective solution for KG-based reasoning. Moreover, in simpler KGs, the single-agent version of R2-KG with strict self-consistency can maintain reliability while further reducing cost.

Limitations

The *Supervisor* makes the final prediction based solely on the triple set and relation list collected by the *Operator*. Consequently, it cannot determine whether the retrieved information is minimal or exhaustive. In multi-label QA tasks, this limitation may cause underprediction, where the framework generates fewer answers than the actual number of correct labels. Additionally, if a query can be answered through multiple relation paths, the *Supervisor* may provide an answer as long as one valid path exists, potentially overlooking alternative correct paths. One way to mitigate this would be to involve the *Supervisor* in every iteration step, but this would remove the distinction between the *Operator* and *Supervisor* roles, increasing computational costs. These constraints stem from the trade-off between cost-effectiveness and reasoning efficiency. While the current design optimizes resource usage, it may not always capture all possible answers in complex reasoning scenarios.

Ethical Consideration

LLM-based KG reasoning requires substantial computational resources, which can contribute to environmental concerns. While our study proposes methods to reduce overall LLM usage, the reliance on large-scale models remains a consideration in terms of environmental impact.

Acknowledgements

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants (No.RS-2019-II190075, No.RS-2022-II220984, No.RS-2024-00338140) and National Research Foundation of Korea (NRF) grants (NRF-2020H1D3A2A03100945), funded by the Korea government (MSIT).

References

Alfonso Amayuelas, Kyle Wong, Liangming Pan, Wenhu Chen, and William Yang Wang. 2024. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6416–6432, Bangkok, Thailand. Association for Computational Linguistics.

Akari Asai and Eunsol Choi. 2021. Challenges in information-seeking QA: Unanswerable questions and paragraph retrieval. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1492–1504, Online. Association for Computational Linguistics.

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

Jan Buchmann, Xiao Liu, and Iryna Gurevych. 2024. Attribute or abstain: Large language models as long document assistants. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8113–8140, Miami, Florida, USA. Association for Computational Linguistics.

Jeremy Cole, Michael Zhang, Daniel Gillick, Julian Eisenschlos, Bhuwan Dhingra, and Jacob Eisenstein. 2023. Selectively answering ambiguous questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 530–543, Singapore. Association for Computational Linguistics.

Shangbin Feng, Weijia Shi, Yike Wang, Wenxuan Ding, Vidhisha Balachandran, and Yulia Tsvetkov. 2024. Don't hallucinate, abstain: Identifying LLM knowledge gaps via multi-LLM collaboration. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14664–14690, Bangkok, Thailand. Association for Computational Linguistics.

Yifu Gao, Linbo Qiao, Zhigang Kan, Zhihua Wen, Yongquan He, and Dongsheng Li. 2024. Two-stage generative question answering on temporal knowledge graph using large language models. *Preprint*, arXiv:2402.16568.

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

Yixin Ji, Kaixin Wu, Juntao Li, Wei Chen, Mingjie Zhong, Xu Jia, and Min Zhang. 2024. Retrieval and reasoning on KGs: Integrate knowledge graphs into large language models for complex question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7598–7610, Miami, Florida, USA. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *Preprint*, arXiv:2305.09645.

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

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023a. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9410–9421, Singapore. Association for Computational Linguistics.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023b. FactKG: Fact verification via reasoning on knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16190–16206, Toronto, Canada. Association for Computational Linguistics.

Yejin Kim, Eojin Kang, Juae Kim, and H. Howie Huang. 2024. Causal reasoning in large language models: A knowledge graph approach. *Preprint*, arXiv:2410.11588.

Hannah Rose Kirk, Andrew M. Bean, Bertie Vidgen, Paul Röttger, and Scott A. Hale. 2023. The past, present and better future of feedback learning in large language models for subjective human preferences and values. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2409–2430, Singapore. Association for Computational Linguistics.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Complex knowledge base question answering: A survey. *Preprint*, arXiv:2108.06688.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.

Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. 2024. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *Preprint*, arXiv:2410.13080.

Jie Ma, Zhitao Gao, Qi Chai, Wangchun Sun, Pinghui Wang, Hongbin Pei, Jing Tao, Lingyun Song, Jun Liu, Chen Zhang, and Lizhen Cui. 2024a. Debate on graph: a flexible and reliable reasoning framework for large language models. *Preprint*, arXiv:2409.03155.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2024b. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. *Preprint*, arXiv:2407.10805.

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

Meta. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Mistral. 2025. Mistral small 3. Mistral AI Research.

OpenAI. 2024a. Gpt-4o mini: advancing cost-efficient intelligence. OpenAI Research.

OpenAI. 2024b. Gpt-4o system card. OpenAI Research.

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

Junwoo Park, Youngwoo Cho, Haneol Lee, Jaegul Choo, and Edward Choi. 2021. Knowledge graph-based question answering with electronic health records. In *Machine Learning for Healthcare Conference*, pages 36–53. PMLR.

Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. 2009. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45.

Qwen. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Apoorv Saxena, Soumen Chakrabarti, and Partha Talukdar. 2021. Question answering over temporal knowledge graphs. *Preprint*, arXiv:2106.01515.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online. Association for Computational Linguistics.

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. *Preprint*, arXiv:2307.07697.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: Skeleton-based semantic parsing for complex questions over knowledge bases. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8952–8959.

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

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. 2025. Paths-over-graph: Knowledge graph empowered large language model reasoning. *Preprint*, arXiv:2410.14211.

Christian Tomani, Kamalika Chaudhuri, Ivan Evtimov, Daniel Cremers, and Mark Ibrahim. 2024. Uncertainty-based abstention in llms improves safety and reduces hallucinations. *Preprint*, arXiv:2404.10960.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Yuqi Wang, Boran Jiang, Yi Luo, Dawei He, Peng Cheng, and Liangcai Gao. 2024. Reasoning on efficient knowledge paths:knowledge graph guides large language model for domain question answering. *Preprint*, arXiv:2404.10384.

Bingbing Wen, Bill Howe, and Lucy Lu Wang. 2024a. Characterizing LLM abstention behavior in science QA with context perturbations. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3437–3450, Miami, Florida, USA. Association for Computational Linguistics.

Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. 2024b. Know your limits: A survey of abstention in large language models. *Preprint*, arXiv:2407.18418.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Kang Liu, and Jun Zhao. 2024. Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering. *Preprint*, arXiv:2404.14741.

Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. 2024. Alignment for honesty. *Preprint*, arXiv:2312.07000.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

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

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2023. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *Preprint*, arXiv:2210.00063.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2017. Variational reasoning for question answering with knowledge graph. *Preprint*, arXiv:1709.04071.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2024. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *Preprint*, arXiv:2305.13168.

A Performance on 1-Hop and 2-Hop Questions

The MetaQA dataset consists of 1-hop, 2-hop, and 3-hop questions; however, our experiments focused exclusively on 3-hop questions. Given that the KG of MetaQA is relatively small and that 1-hop and 2-hop questions are considerably simpler than 3-hop questions, we excluded them from our primary evaluation. Nevertheless, to assess the R2-KG across different levels of task complexity, we randomly sampled 100 questions from 1-hop and 2-hop sets and evaluated the performance. As shown in Table 8, R2-KG exhibited strong performance with high coverage.

B Examples of the Two Excluded Question Types in CRONQUESTIONS

Unlike the four other datasets, CRONQUESTIONS is constructed with a five-element KG, where each quintuple follows the format: [head, relation, tail, start time, end time]. This structure includes temporal information, specifying the start and end years of an event. CRONQUESTIONS contains five types of reasoning tasks: Simple time, Simple entity, Before/After, First/Last, and Time Join. However, in our experiments, we excluded the Before/After and First/Last question types. The primary reason is that, while our framework predicts answers based on the KG, these question types often contain subjective ground truth labels that do not fully align with the available KG information. For example, this is a sample of Before/After question: "Which team did Roberto Baggio play for before the Italy national football team?" Using our framework, we can retrieve the following KG facts related to Roberto Baggio: [Roberto Baggio, member of sports team, ACF Fiorentina, 1985, 1990] [Roberto Baggio, member of sports team, Brescia Calcio, 2000, 2004] [Roberto Baggio, member of sports team, Vicenza Calcio, 1982, 1985] [Roberto Baggio, member of sports team, Juventus F.C., 1990, 1995] [Roberto Baggio, member of sports team, Italy national football team, 1988, 2004] [Roberto Baggio, member of sports team, A.C. Milan, 1995, 1997] [Roberto Baggio, member of sports team, Bologna F.C. 1909, 1997, 1998]. According to the KG, Roberto Baggio joined the Italy national football team in 1988. Before that, he played for Vicenza Calcio (starting in 1982) and ACF Fiorentina (starting in 1985), meaning both teams are valid answers. However, the ground truth

Model	Parameter	Architecture
Qwen2.5-14B	14.7B	transformers with RoPE, SwiGLU, RMSNorm, and Attention QKV bias
Qwen2.5-32B	32B	transformers with RoPE, SwiGLU, RMSNorm, and Attention QKV bias
Llama-3.1-70B	70B	auto-regressive language model that uses an optimized transformer architecture.
Mistral-Small-Instruct-2409	22B	Unknown
GPT-4o mini, GPT-4o	Unknown	Unknown

Table 7: Specification of models

Dataset	Metrics			
	Coverage	F1 (M)	F1 (S)	Hit
MetaQA 1-hop	100	95.3	98.9	100
MetaQA 2-hop	96.0	99.9	99.7	100

Table 8: Performance of R2-KG in MetaQA 1-hop and 2-hop. F1 (M): Micro F1 score, F1 (S): Samplewise F1 score.

label in the dataset only includes ACF Fiorentina, omitting Vicenza Calcio, despite it being a correct answer based on the KG. Due to this labeling inconsistency, objective evaluation of these question types becomes unreliable. As a result, we decided to exclude these types from our experiments.

C R2-KG vs. baselines with various LLMs as *Operator* and *Supervisor*

To demonstrate that R2-KG consistently outperforms the baseline regardless of the underlying LLM's capability, we conducted the following experiment. In the dual-agent setup, we varied not only the *Operator* but also the *Supervisor* across three different LLMs(LLMs—Mistral-Small, GPT-4o mini, and GPT-4o). The results show that even when the *Supervisor* is a low-capacity LLM, R2-KG still achieves higher F1 scores and hit rates than the baseline. As shown in Table 12, on both the FactKG and MetaQA 3-hop datasets, R2-KG consistently outperforms ToG by a substantial margin. In FactKG, when both the *Operator* and *Supervisor* use GPT-4o mini, R2-KG achieves a hit rate of 89.1%, whereas ToG with GPT-4o reaches only 86.8%. This clearly demonstrates that even when using a low-capacity model, R2-KG surpasses the baseline equipped with a high-capacity model. Similarly, on MetaQA 3-hop, R2-KG outperforms ToG by nearly 30% in terms of hit rate, further confirming its robustness and efficiency across model scales.

D Prompt Structure and Usage

Each prompt for *Operator* consists of three components: Task description, Helper function explanations, and three few-shot examples. When using R2-KG, users only need to modify the few-shot examples to match the specific dataset while keeping the rest of the prompt unchanged. Examining the prompts reveals that when the *Operator* requests a helper function, the *Operator* can request multiple instances in a single iteration based on its needs. Additionally, it can request different types of functions simultaneously. The prompt for *Supervisor* contains the following elements: Task description, triples collected so far by the *Operator*, a relation list for each entity, and few-shot examples. The reason for explicitly including the entity-wise relation list is to ensure that when the *Supervisor* provides feedback to the *Operator*, it requests subgraphs that actually exist in the KG. During pilot testing, when the relation list was not provided, the system occasionally requested non-existent entity-relation pairs in the KG. This resulted in ineffective feedback and ultimately failed to assist the *Operator* in its KG exploration.

E Final Reasoning Path Construction

When sufficient evidences are given from *Operator* to *Supervisor*, then the *Supervisor* selects the necessary triples and constructs a final reasoning path that aligns with the claim structure. Assume that the given query is "Which languages were used in the films directed by the same directors as [The Vanishing American]" and G_k given by the *Operator* are as follows (tilde (~) represents the inverse direction of relation): [The Vanishing American, directed_by, George B. Seitz], [George B. Seitz, ~directed_by, The Last of the Mohicans], [George B. Seitz, ~directed_by, Love Finds Andy Hardy], [The Last of the Mohicans, in_language, English], [Love Finds Andy Hardy, in_language, French]. Then, *Supervisor* generates two final reasoning paths: (The Vanishing American-George B. Seitz-The Last of the Mohicans-English), and

Reasoning Path Strategy	Utilized Model		WebQSP				MetaQA 3-hop				CRONQUESTIONS				FactKG	
	Operator	Supervisor	Coverage	F1 (M)	F1 (S)	Hit	Coverage	F1 (M)	F1 (S)	Hit	Coverage	F1 (M)	F1 (S)	Hit	Coverage	Hit
Multi Prompts	GPT-4o mini	GPT-4o	69.2	77.1	85.5	93.6	89.2	92.3	95.4	**100**	83.6	**34.5**	**85.1**	98.6	56.3	94.5
Paraphrasing	GPT-4o mini	GPT-4o	70.7	73.8	85.7	**93.9**	87.3	**92.5**	**95.9**	99.9	84.0	33.5	84.2	98.4	55.0	93.6
Top-p / Temperature	GPT-4o mini	GPT-4o	73.6	**81.2**	**86.0**	92.8	87.1	92.4	95.8	**100**	23.9	27.8	77.4	**99.7**	16.6	**95.2**

Table 9: Performance of dual-agent version of R2-KG with self-consistency on the four KG-based reasoning benchmarks. We denote the **best** method for each metric (except coverage). F1 (M): Micro F1 score, F1 (S): Samplewise F1 score.

Utilized Model		WebQSP				MetaQA 3-hop				CRONQUESTIONS				FactKG	
Operator	Supervisor	Coverage	F1 (M)	F1 (S)	Hit	Coverage	F1 (M)	F1 (S)	Hit	Coverage	F1 (M)	F1 (S)	Hit	Coverage	Hit
GPT-4o mini	GPT-4o mini	85.0	78.8	79.9	88.3	65.7	92.8	95.7	98.6	84.9	30.5	81.8	98.8	78.4	93.0

Table 10: Performance of baselines and R2-KG on the four KG-based reasoning benchmarks on the entire test set. F1 (M): Micro F1 score, F1 (S): Samplewise F1 score.

Method	Utilized Model		Synthetic MetaQA 3-hop			
	Operator	Supervisor	Cvg	F1 (M)	F1 (S)	Hit
KG-GPT	Mistral-Small	–	100	4.5	16.1	42.0
	GPT-4o mini	–	99.0	10.8	27.0	94.9
	GPT-4o	–	99.0	10.9	27.8	96.0
ToG	Mistral-Small	–	19.5	3.3	7.8	28.2
	GPT-4o mini	–	15.0	14.2	19.5	70.0
	GPT-4o	–	14.5	15.3	21.1	89.7
R2-KG	Qwen2.5-14B	GPT-4o	88.0	89.1	89.6	93.8
	Qwen2.5-32B	GPT-4o	90.5	**98.4**	98.8	99.4
	Mistral-Small	GPT-4o	58.5	92.4	93.4	94.9
	Llama-3.1-70B	GPT-4o	87.0	94.8	96.9	98.9
	GPT-4o mini	GPT-4o	87.5	93.0	94.1	96.6
	GPT-4o	GPT-4o	97.0	98.1	**99.1**	**100**

Table 11: Performance using synthetic KG (modified KG based on MetaQA). We denote the **best** method for each metric (except coverage). Cvg: Coverage, F1 (M): Micro F1 score, F1 (S): Samplewise F1 score.

(The Vanishing American-George B. Seitz-Love Finds Andy Hardy-French). Finally, *Supervisor* generates an answer and returns to *Operator* (*i.e.*, English, French in the given example).

F Model Spec

Please check Table 7. Entries labeled as "Unknown" indicate that the information is not publicly available.

G Discussion for the Low Metric Scores for WebQSP, CWQ, and CRONQUESTIONS

Based on the result shown in Table 2, R2-KG showed slightly lower performance in the micro F1 score, sample-wise F1 score, and hit rate for WebQSP and CWQ (< 90%), and micro F1 score for CRONQUESTIONS (< 50%). Through insights from the original paper and our empirical experiments, we observed that these datasets exhibit inherent limitations. This suggests the existence of upper bounds that make achieving 100% performance infeasible even though they are used in many previous research.

FactKG			
Method	**Model**	**Cvg**	**Hit**
ToG	Mistral-Small	52.8	69.5
ToG	GPT-4o mini	35.8	83.5
ToG	GPT-4o	50.6	86.8
R2-KG	Mistral-Small	56.2	82.9
R2-KG	GPT-4o mini	78.4	89.1
R2-KG	GPT-4o	77.8	93.1

MetaQA 3-hop					
Method	**Model**	**Cvg**	**F1(M)**	**F1(S)**	**Hit**
ToG	Mistral-Small	24.1	13.2	31.2	62.2
ToG	GPT-4o mini	30.5	13.6	28.5	67.2
ToG	GPT-4o	24.5	15.6	44.0	95.5
R2-KG	Mistral-Small	79.0	77.9	84.3	92.7
R2-KG	GPT-4o mini	65.7	92.8	95.7	98.6
R2-KG	GPT-4o	98.3	98.3	99.2	99.9

Table 12: Results on FactKG and MetaQA 3-hop benchmarks. R2-KG employs both *Operator* and *Supervisor* agents with three different models.

In the original WebQSP paper (Yih et al., 2016), experts manually annotated 50 sample questions by constructing corresponding SQL queries. The reported correctness of these annotations ranged from 92% to 94%, indicating that even human-generated queries did not achieve perfect accuracy. This highlights the intrinsic difficulty of achieving complete evidence collection for every query in the WebQSP dataset. Although the original paper of

CWQ does not provide a quantitative correctness analysis, the dataset is derived from WebQSP and inherits similar structural issues.

Through our own experiments, we observed several such limitations in both WebQSP and CWQ. The most critical issue was that, in some cases, a correct answer (other than the ground-truth label) entity could be retrieved using a semantically equivalent but different relation than the one used in the annotated query; however, such answers were not included in the ground-truth labels. Other minor issues included mismatches between the recorded entity and the actual answer, and inconsistencies between the question and the label (*e.g.,* questions asking for "two states" but only one state being annotated for the label).

These factors collectively suggest that achieving 100% performance in terms of F1 scores or hit rate is practically infeasible on these datasets. As a result, the slightly lower scores we observe can be attributed, at least in part, to these dataset-inherent limitations rather than model deficiencies alone.

CRONQUESTIONS exhibits high variance in the number of labels per question—some questions have a single answer, while others require many. As the number of label of the question increases, the corresponding subgraph lengthens, leading to token length issue or prediction failures. Even when the reasoning path is correct, covering all labels becomes challenging, which substantially impacts the micro F1 score. However, as seen from the sample-wise F1—where CRONQUESTIONS still achieves 89%—R2-KG generally demonstrates strong reasoning capability even under such challenging condition.

H R2-KG Dual-Agent Approach Combined with Self-Consistency Strategy

Table 9 shows the result of combining the self-consistency strategy with the dual-agent approach. The *Supervisor* generated the final answer based on three trials, leading to stricter predictions. As a result, coverage was lower compared to using dual-agent R2-KG alone. For WebQSP and MetaQA, the F1 score was lower than that of a single trial of the dual-agent R2-KG, whereas the hit rate was significantly higher. This is because, applying the strict self-consistency technique, some multi-label predictions were filtered out, meaning the model did not perfectly match all ground truth labels but still correctly predicted at least one. For CRONQUESTIONS, coverage, F1 scores, and hit rate were relatively lower. This dataset contains a significantly higher number of ground truth labels than others, making it difficult for any single trial to cover all labels. Consequently, the final prediction lacked sufficient labels. In FactKG, coverage varied widely, ranging from 10% to 50% depending on the reasoning path method. However, the hit rate consistently remained above 93%, indicating strong performance. Overall, for multi-label tasks with many ground truth labels, a single trial using R2-KG: a single trial of dual-agent approach performed more effectively than dual-agent with self-consistency strategy, suggesting that dual-agent with self-consistency strategy is not always beneficial for complex multi-label reasoning tasks.

I R2-KG Using the Full Dataset

Table 10 presents the results obtained using the full dataset. In this experiment, both the *Operator* and *Supervisor* were set to GPT-4o mini, and the experimental setup remained identical to the main experiment.

For MetaQA 3-hop, CRONQUESTIONS, and FactKG, the hit rate exceeded 90%, with CRONQUESTIONS reaching an impressive 98.6%. However, coverage was generally lower or similar compared to the main experiment. This decline is likely due to the *Supervisor*'s limited ability to construct the correct reasoning path using the triple set during inference, as it was replaced with GPT-4o mini instead of GPT-4o. Although sufficient evidence was available, the *Supervisor* struggled to appropriately combine the necessary components of the query, leading to failed predictions. These results further highlight the critical role of the *Supervisor* in the reasoning process.

Despite the slight performance drop compared to the main experiment due to the relatively low-capacity *Supervisor*, the framework still significantly outperforms baseline methods. The effectiveness of the *abstention mechanism* remains evident, ensuring that the system generates reliable predictions while maintaining robustness against uncertainty.

J Qualitative Analysis

Figure 4 illustrates an example where R2-KG successfully performs reasoning on WebQSP, while Figure 5 shows a case where it fails. Within the

T of 15, each box represents the *Operator*'s reasoning (gray), the *Server*'s execution result (blue), and the *Supervisor*'s reasoning (red). Some parts of the iteration process have been omitted due to excessive length.

K Details of Single-Agent Version of R2-KG combined with Self-Consistency Strategy

The typical Self-Consistency strategy allows the language model to generate multiple reasoning paths and selects the most common answer across all trials. In contrast, our approach applies a stricter criterion, selecting the final answer only when all trials reach unanimous agreement. The details of various reasoning paths to generate multiple responses are as follows; The prompt used for the single-agent approach, where the *Operator* handles both KG retrieval and answer generation, is shown in Figure 8. For the Multi-Prompt approach, the same base prompt was used, with only the few-shot examples adjusted for in-context learning. The prompt used for query paraphrasing is identical to that in Figure 9. In this approach, each query is rewritten into three semantically equivalent but structurally different forms, and each variation is processed independently by a low-capacity LLM for three reasoning attempts. The parameter combinations used for LLM Top-p / Temperature variation are as follows: (Top-p, Temperature) = (0.3, 0.5), (0.7, 1.0), (0.95, 0.95)

L LLM Usage Statistics and Latency Analysis

KG-GPT requires at least 3 calls (*i.e.*, Sentence Segmentation, Graph Retrieval, and Inference) to a high-capacity LLM, and ToG makes a minimum of 4 and maximum of 25 such requests, depending on the number of reasoning paths, which is closely related to the depth and width limit (*i.e.*, hop limit, beam search width limit in KGs) used for ToG hyperparameter. This represents a substantial reduction compared to R2-KG 's invocation of high-capacity LLMs fewer than 1.43 times per query. In other words, R2-KG dramatically lowers model usage costs while ensuring superior performance. One might also worry about added latency from *Supervisor –Operator* interactions; however, measuring per-query inference times for the single-agent and dual-agent versions reveals only a 0.97s difference, which we confirm is not practically problematic.

M Token Efficiency and Cost Analysis

In Section 7.3, we presented the token usage data, which is summarized in Table 5. This section provides a detailed analysis of the associated token costs.

As of July 28, 2025, under OpenAI's pricing policy GPT-4o-mini charges USD 0.15 (input) + USD 0.60 (output) per 1M tokens, and GPT-4o charges USD 2.50 (input) + USD 10.00 (output) per 1M tokens. Therefore, on CWQ the single-agent version (19,309 tokens with GPT-4o) costs about USD 0.0483 and the dual-agent version (18,987 tokens with GPT-4o-mini + 4,912 tokens with GPT-4o) costs USD 0.0153 per query. In this way, by clearly separating the Operator and Supervisor roles, we achieve a USD 0.033 per query saving compared to using a high-capacity LLM for the entire process.

N Experimental Setting for Baselines

Among the baselines used in the experiment, ToG allows width and depth to be set as hyperparameters. In our experiments, the depth was set to 3 for all datasets except FactKG, where it was set to 4. By default, ToG's width is set to 3, meaning it considers up to three entities or relations per step, regardless of the type of subject. However, this setting was highly ineffective for multi-label tasks. To improve its performance, we separately configured (relation-width, entity-width) to optimize results. The values used in the main experiment were as follows: FactKG, MetaQA, and WebQSP were set to (3, 7), (2, 5), and (3, 3), respectively.

Additionally, when ToG fails to retrieve supporting evidence from the KG, it generates answers based on the LLM's internal knowledge. To ensure a fair comparison based solely on KG-derived information, we treated cases where ToG relied on internal knowledge after KG retrieval as *Abstentions*. Similarly, while KG-GPT does not have a built-in *abstention mechanism*, we considered instances where the model failed to generate a final answer due to errors during its three-step process (sentence segmentation, graph-retrieval, and inference)—such as token length limits or parsing failures—as *Abstentions*.

For both baselines, prompt tuning was conducted to align them with each dataset. Specifically, we modified the few-shot examples extracted from

Original	**Modified (Contradictory)**
`[Inception, directed_by, Christopher Nolan]`	`[Inception, directed_by, Mario Van Peebles]`
Q: Who is the director of *Inception*?	**Q:** Who is the director of *Inception*?
A: Christopher Nolan	**A:** Mario Van Peebles

Table 13: Original vs. contradictory triples and QA pairs (MetaQA)

each dataset while keeping the default prompt structure unchanged.

O Experiment for Measuring Knowledge Shortcut for R2-KG and Baselines

We further carried out an experiment to determine whether the LLM answers by drawing on its pre-trained world knowledge (internal knowledge) or by relying on the subgraph it has explored. To make this distinction, we built a synthetic KG that contradicts commonly accepted facts and ran R2-KG on it. We constructed the synthetic KG/queries as follows based on MetaQA as shown in Table 13.

As shown in Table 11, R2-KG maintains high F1 score and hit rates even when evaluated on a KG that deliberately contradicts real-world facts, along with corresponding queries. This confirms that when a subgraph is provided, the LLM bases its reasoning on the retrieved structure rather than latent, pre-trained knowledge. In contrast, ToG shows extremely low accuracy, as it relies on internal knowledge when its KG exploration fails. These results demonstrate that R2-KG not only explores KGs more effectively, but also grounds its answers in retrieved evidence—substantially mitigating knowledge shortcuts.

	Qid : WebQTest-62 Question : what was jesse james killed with? GT entity : ('Jesse James', 'm.02khkd')
Iteration	
0	Statement: First, I need to find information related to Jesse James. I'll start by checking the relations linked to Jesse James. Helper function: getRelation['Jesse James']
	Execution result: Relations_list["Jesse James"] = ['people.person.date_of_birth', 'user.alexander.misc.murdered_person.date_murdered', 'common.topic.description', 'base.popstra.celebrity.dated', 'people.person.parents', 'base.popstra.celebrity.friendship', 'common.topic.topic_equivalent_webpage', 'common.topic.webpage', 'common.topic.notable_types', 'people.person.place_of_birth', 'type.object.name', 'people.person.sibling_s', 'en', 'common.topic.image', 'common.topic.notable_for', 'common.topic.alias', 'user.alexander.misc.murdered_person.murder_method', …
1	Statement: To find out how Jesse James was killed, I will explore the graph for the relations related to the murder method. Helper function: exploreKG['Jesse James']=['user.alexander.misc.murdered_person.murder_method']
	Execution result: ['Jesse James', 'user.alexander.misc.murdered_person.murder_method', 'Assassination']
2	Statement: Jesse James was killed by assassination. Therefore, the answer to the claim about what Jesse James was killed with is 'Assassination'. Helper function: Verification[]
	Execution result: The triple set provides information that Jesse James was murdered using the method 'Assassination'. However, the claim specifically asks for the weapon or tool used in the killing, which is not explicitly provided by the given relation 'user.alexander.misc.murdered_person.murder_method'. Therefore, the current data is insufficient to determine the exact weapon used.
3 ~ 6	Statement: Since the information I received only specifies that Jesse James was killed by assassination and does not provide details about the weapon used, I need to gather more information to find out what specific weapon or tool was involved in the assassination. I'll check for any relations that might give details about the weapon used in the murder. Helper function: getRelation['Jesse James']
	Execution result: Relations_list["Jesse James"] = ['people.person.date_of_birth', 'user.alexander.misc.murdered_person.date_murdered', 'common.topic.description', 'base.popstra.celebrity.dated', 'people.person.parents', 'base.popstra.celebrity.friendship', ….
7	Statement: Jesse James was murdered at the Jesse James Home Museum. However, this information does not provide insights into the specific weapon used in the assassination. As I have not yet found the necessary details regarding the weapon, I will try exploring other relations linked to Jesse James that may provide clues about the murder. Helper function: exploreKG['Jesse James']=['people.deceased_person.cause_of_death']
	Execution result: ['Jesse James', 'people.deceased_person.cause_of_death', 'Assassination'], ['Jesse James', 'people.deceased_person.cause_of_death', 'Firearm']
8	Statement: The evidence indicates that Jesse James's cause of death was classified as 'Assassination' and specifically involved a 'Firearm'. This suggests that a firearm was the weapon used in his assassination. Therefore, the answer to the claim about what Jesse James was killed with is 'Firearm'. Helper function: Verification[]
	Execution result: Done!! predict : ['Firearm'], gt_label:['Firearm']

Figure 4: Successful Case in WebQSP. *Supervisor* effectively guides the model to extract a more relevant answer for the question. *Operator*, *Server* Response, *Supervisor* for each colored box.

	Qid : WebQTest-142 Question : who developed the tcp ip reference model? GT entity : ('Transmission Control Protocol','m.07hzk')
Iteration	
0	Statement: First, I need to find information about the Transmission Control Protocol to identify its relationships and any relevant connections to the TCP/IP reference model. Helper function: getRelation['Transmission Control Protocol']
	Execution result: Relations_list["Transmission Control Protocol"] = ['common.topic.description', 'common.topic.topic_equivalent_webpage', 'common.topic.notable_types', 'type.object.name', 'en', 'common.topic.image', 'common.topic.notable_for', 'common.topic.alias', 'type.object.type', 'computer.internet_protocol.software_used_by', 'common.topic.article', 'base.argumentmaps.original_idea.innovator', 'base.ontologies.ontology_instance.equivalent_instances', 'user.avh.default_domain.ellerdale_topic.ellerdale_id', 'law.invention.inventor', 'type.object.key', 'base.schemastaging.context_name.pronunciation', 'user.avh.ellerdale', 'kg.object_profile.prominent_type']
1	Statement: To find out who developed the TCP/IP reference model, I need to explore the graph with a relation that may link to its inventor or origin. The relation that seems most relevant is 'base.argumentmaps.original_idea.innovator' as it could lead to the innovator of the TCP/IP model. Helper function: exploreKG['Transmission Control Protocol']=['base.argumentmaps.original_idea.innovator']
	Execution result: ['Transmission Control Protocol', 'base.argumentmaps.original_idea.innovator', 'Robert E. Kahn'], ['Transmission Control Protocol', 'base.argumentmaps.original_idea.innovator', 'Vint Cerf']
2	Statement: The evidence shows that the TCP/IP reference model was developed by Robert E. Kahn and Vint Cerf. Therefore, the answer to the claim about who developed the TCP/IP reference model is both 'Robert E. Kahn' and 'Vint Cerf'. Helper function: Verification[]
	Execution result: The triple set indicates that the 'Transmission Control Protocol' was developed by 'Robert E. Kahn' and 'Vint Cerf' using the relation 'base.argumentmaps.original_idea.innovator'. However, it does not explicitly state whether these individuals also developed the 'TCP/IP reference model' as a whole. The relations provided for 'Transmission Control Protocol' do not include any specific relation indicating its connection to the 'TCP/IP reference model'. Hence, we lack sufficient evidence to confirm that these individuals developed the 'TCP/IP reference model'.
3	Statement: Since we need more information to verify the direct connection between Robert E. Kahn, Vint Cerf, and the development of the TCP/IP reference model, I will look for additional evidence regarding either Robert E. Kahn or Vint Cerf that explicitly links them to the TCP/IP reference model. I will start by finding relations associated with Robert E. Kahn. Helper function: getRelation['Robert E. Kahn']
	Execution result: You gave wrong format of getRelation[] function. Follow the format of examples.
4	Statement: I need to gather information about Robert E. Kahn to find relevant connections to the TCP/IP reference model. Helper function: getRelation['m.0k4q8'] ## getRelation['m.0k4q9']
	Execution result: You gave wrong format of getRelation[] function. Follow the format of examples.
5 ~ 14	Statement: I apologize for the error. I will first find the relations linked to Robert E. Kahn to gather more information. Helper function: getRelation['Robert E. Kahn']
	Execution result: You gave wrong format of getRelation[] function. Follow the format of examples.
Exceed Limit	"Abstain"

Figure 5: Failure Case in WebQSP. *Supervisor* fails to infer, leading the *Operator* to invoke functions in the wrong format repeatedly. *Operator*, *Server* Response, *Supervisor* for each colored box.

Prompt for Operator

Your task is finding proper labels for given claim based on the graph data without your base knowledge.
You can use below helper functions to find the evidence for finding labels.
<Helper Functions>
1.getRelation[entity]: Returns the list of relations linked to the entity. You can choose several relations from the list that seem related to the claim.
2.exploreKG[entity]=[relation_1,relation_2, ... relation_K]: Returns the triple set around the entity. For example, [entity, relation_2, tail entity] etc. You can choose relation from [User]'s execution result.
3.Verification[]: After getting enough evidence after exploreKG() helper function and if verification can be done, call this function. If [User] requires more information, you need to collect more triples in following steps.

You must follow the exact format of the given helper function. Now, I will give you a claim and Given Entity that you can refer to. However, some of the entities needed in verification are not included in Given Entity. You have to use proper helper functions to find proper information to verify the given claim. Once you give a response about helper function, stop for [User] response. If response has made, continue your [Your Task] (Do not make multiple 'Helper function: ' lines). Importantly, Do not change the format of the entity or relation including '~'.
Here are some examples.

<3 Few-shot Examples>
(Example 1)
Question : Who was district attorney when J. D. Rees was the Member of the 31st Parliament of the United Kingdom
Given entity : ['district attorney', 'Member of the 31st Parliament of the United Kingdom', 'J. D. Rees']

[Your Task]
Statement : Let's see what relations linked to each entity, 'district attorney', 'Member of the 31st Parliament of the United Kingdom', 'J. D. Rees'.
Helper function : getRelation['district attorney'] ## getRelation['Member of the 31st Parliament of the United Kingdom'] ## getRelation['J. D. Rees']
[User]
Execution result :
Relation_list['district attorney']=['~position held', '~occupation']
Relation_list['Member of the 31st Parliament of the United Kingdom']=['~position held']

…
Now, it's your turn.
Claim : <<<Question>>>
Given entity: <<<Entity set>>>

Let's start the process.

Figure 6: Used for FactKG. [Your Task] is generated by the *Operator*, while [User] represents either the *Server*'s response or the *Supervisor*'s answer.

Prompt for Supervisor

You are the evaluator. I will show you a claim and a triple set extracted from a graph. Based on the given triple set and relation list of each entity, determine whether the claim is True or False.
If given triple sets are lack of information to verify the claim, give the the combination of entity and relation you need. You can refer the given relations list and choose what relation information is more needed.
The triple set takes the form [Head, Relation, Tail], which means 'Head's Relation is Tail.' If the relation starts with '~', it indicates a reverse relation, meaning 'Tail's relation is head.'
The following cases may arise: Choose one option from 'Executable (True or False)' or 'Not executable(Insufficient evidence)'.
If you choose 'Not executable(Insufficient evidence)', You must specify in the statement which additional relation information is needed for a particular entity. However, the relation can only be selected from the given Relation_list and cannot be created arbitrarily.
Refer to the explanations of the two options below to answer the Statement and Evaluation.

<Cases>
1. If the triple sets are sufficient to determine the True/False of the claim --> Executable (True or False)
2. If the triple set is insufficient or ambiguos to determine the True/False of the claim --> Not executable (Insufficient evidence)

<8 Few-shot examples>
(Example 1)
[User]
Claim : A fictional character, which was created by Joe Quesada, is broadcast on Lebanese Broadcasting Corporation and starred Tim Brooke-Taylor.
Triple sets : [['Joe_Quesaa', '~creator', 'Azrael_(comics)'], ['Joe_Quesaa', '~creator', 'Menace_(Marvel_Comics)'] …
Relations of Entity :
Relations_list["Tim_Brooke-Taylor"] = ['birthPlace', 'honorificSuffix', '~after', 'notableWork', 'genre', 'years', 'spouse', 'title', 'givenName', 'shortDescription', 'surname'…]
[Your Task]
Statement : We need more information for verification. Try to look relation '~starring' linked with Tim_Brooke-Taylor and relation 'broadcastArea, ~channel,~tv' linked with Lebanese_Broadcasting_Corporation.
Evaluation : Not executable (Insufficient evidence)
…
Now, it's your turn. Continue your 'Statement and Evaluation' task. Response must have same form with upper examples.
[User]
Claim : <<<<CLAIM>>>>
Triple set : <<<<Triple set>>>>
Relations of Entity : <<<<GOLD RELATIONS>>>>

Figure 7: Used for FactKG. [Your Task] is generated by the *Operator*, while [User] contains the given query and the evidence collected by the *Operator*.

Prompt for Single-Agent version

Your task is finding proper labels for given claim based on the graph data without your base knowledge.
You can use one of the helper functions below to find the evidence for finding labels.

<Helper Functions>
1.getRelation[entity]: Returns the list of relations linked to the entity. You can choose several relations from the list that seem related to the claim.
2.exploreKG[entity]=[relation_1,relation_2, ... relation_K]: Returns the triple set around the entity. For example, [entity, relation_1, tail entity] etc. You can choose relation from [User]'s execution result.
3.Verification[True/False]: If you can judge the claim as True or False, give the answer.

You must follow the exact format of the given helper function.

Now, I will give you a claim and Given Entity that you can refer to.
However, some of the entities needed in verification are not included in Given Entity.
You have to use proper helper functions to find proper information to verify the given claim.
Once you give a response about helper function, stop for my response. If response has made, continue your 'Statement and Helper function' task.
Importantly, Do not change the format of the entity or relation including '~'.

<3 Few shot Examples>
Example 1)
Claim: Yea he was born in Zaoyang, Hubei
Given entity: ["Zaoyang", "Hubei"]

[Your Task]
Statement : First, I need to look around the given entities. I will start by getting the relation list linked to Zaoyang.
Helper function : getRelation["Zaoyang"]
[User]
Execution result :
Relations_list["Zaoyang"] = ['augRecordLowC', 'decLowC', 'aprRecordHighC', 'yearLowC', 'junMeanC', 'unitPrecipitationDays', 'novLowC', 'longm', 'julRecordLowC',…
[Your Task]
Statement : Now, I need to look around the given entities. I will get the relation list linked to Hubei.
Helper function : getRelation["Hubei"]
[User]
Execution result :
Relations_list["Hubei"] = ['popyear', '~placeOfDeath', 'south', '~territory', 'owl#differentFrom', '~routeStart', '~locale', '~capital', '~place', 'nationalities', 'prefectures', 'gdprank',
[Your Task]
Statement : To verify the claim, I need to explore the graph for Zaoyang to find any relation with '~birthPlace' or '~placeOfBirth'.
Helper function : exploreKG["Zaoyang"]=['~birthPlace', '~placeOfBirth']
[User]
Execution result :
['Zaoyang', '~birthPlace', 'Nie_Haisheng'], ['Zaoyang', '~placeOfBirth', 'Nie_Haisheng']
[Your Task]
Statement : According to the triples, Nie Haisheng was born in Zaoyang. However, the claim does not specify who "he" refers to. To verify the claim, I need to check if Zaoyang is part of Hubei. I will explore the graph for Zaoyang to find any relation with '~isPartOf'.
Helper function : exploreKG["Zaoyang"]=['~isPartOf']
[User]
Execution result :
['Zaoyang', '~isPartOf', 'Xiangyang'], ['Zaoyang', '~isPartOf', 'Hubei']
Statement : According to the triples, Zaoyang is part of Hubei. Therefore, the claim that "he was born in Zaoyang, Hubei" is consistent with the information provided. The claim is True.
Helper function : Verification[True]
…
Now, it's your turn. Your response must have same form with upper examples.
Claim: <<<<CLAIM>>>>
Given entity: <<<<GT_ENTITY>>>>

Figure 8: Prompt for single version of R2-KG [Your Task] is generated by the *Operator*, while [User] represents the *Server*'s response.

Algorithm 1: Dual-Agent R2-KG Reasoning Process

Input: Claim c, Given entity e_0, Iteration limit T
Output: Final reasoning result or Abstain if limit exceeded

$E_{\text{seen}} \leftarrow \{e_0\}$ `// Set of seen entities`
$Relations \leftarrow \{\}$ `// Dictionary for entity-relation pairs`
$gold\_triples \leftarrow \{\}$ `// Collected triples`
$chat\_log \leftarrow \{\}$ `// Stored interaction logs`
$i \leftarrow 0$ `// Iteration counter`
while $i < T$ **do**
    $response \leftarrow$ Opeartor_response$(c, E_{\text{seen}}, Relations, chat_log)$
    Append $response$ to $chat\_log$
    if $response =$ *getRelation*(e) **then**
        $list\_of\_relations \leftarrow$ Server_response(e)
        $Relations[e] \leftarrow list\_of\_relations$
        Append $list\_of\_relations$ to $chat\_log$
    else if $response =$ *exploreKG*(e, rel) **then**
        $triples \leftarrow$ Server_response(e, rel)
        $gold\_triples \leftarrow gold\_triples \cup triples$
        Append $triples$ to $chat\_log$
    else if $response =$ *verification*$()$ **then**
        $feedback \leftarrow$ Supervisor_response$(gold_triples, Relations)$
        if $feedback$ *is an answer* **then**
            result $\leftarrow feedback$
            Break
        else
            Append $feedback$ to $chat\_log$
    $i \leftarrow i + 1$
if $i \geq T$ **then**
    Return Abstain
else
    Return result

Prompt for Claim Paraphrasing

You are the paraphraser. Rewrite the following claim in a more formal tone, keeping the core idea intact.
If the given claim has a colloquial style format, paraphrase it into written style format.
Give three paraphrased sentences for given claim.

<8 Few shot Examples>
(Example 1)
[User]
Claim : The airport in Punjab, Pakistan is operated by the government agency of the Jinnah International Airport.
[ChatGPT]
1. The government agency responsible for Jinnah International Airport oversees operations at an airport located in Punjab, Pakistan.
2. An airport in Punjab, Pakistan is managed by the same governmental body that operates Jinnah International Airport.
3. The operations of an airport in Punjab, Pakistan are handled by the government agency managing Jinnah International Airport.

(Example 2)
[User]
Claim : Tim Brooke-Taylor starred as a fictional character, which was first aired on 10/03/1983 and was broadcast by STV.
[ChatGPT]
1. Tim Brooke-Taylor portrayed a fictional character in a show that premiered on October 3, 1983, and was aired by STV.
2. STV broadcasted a program featuring Tim Brooke-Taylor as a fictional character, first aired on October 3, 1983.
3. On October 3, 1983, STV premiered a show in which Tim Brooke-Taylor starred as a fictional character.
…
Now, it's your turn. Continue your paraphrasing task. Response must have same form with upper examples.
[User]
Claim : <<<<CLAIM>>>>

Figure 9: Prompt for query paraphrasing. [User] contains the query to be paraphrased, while [ChatGPT] generates three different variations of the sentence.