GRASP: Graph Agentic Search over Propositions for Multi-hop Question Answering

Stockton Jenkins, Ramya Korlakai Vinayak, Junjie Hu
University of Wisconsin-Madison
{jsjenkins4,korlakaivina,junjie.hu}@wisc.edu

arXiv:2605.16598v1 [cs.MA] 15 May 2026

Abstract

Agentic retrieval improves multi-hop question answering by giving language models autonomy to iteratively gather evidence. Recent work augments these systems with knowledge graphs for structured traversal, but this combination introduces significant cost: expensive graph construction at index time and compounding token usage at inference time. We introduce **Gr**aph **A**gentic **S**earch over **P**ropositions (GRASP), an agentic system that simultaneously optimizes for high accuracy and minimal token usage in multi-hop question answering. Rather than executing a rigid, singular query, GRASP actively coordinates its retrieval strategy by decomposing multi-hop queries into dependency-aware plans. This enables GRASP to dynamically scale the number of sub-agents according to the complexity of the problem. Each sub-agent resolves its single-hop query by exploring a novel three-layer hierarchical graph of entities, propositions, and passages, using the entity layer for targeted traversal and the proposition layer for high-recall passage retrieval via reciprocal-rank voting. We evaluate GRASP on MuSiQue, 2WikiMultihopQA, and HotpotQA under two settings: open-corpus retrieval and extended context reasoning (LongBench). GRASP achieves the highest QA accuracy in the open retrieval setting on MuSiQue and 2Wiki while using **40–50% fewer tokens** than IRCoT+HippoRAG2. Furthermore, GRASP leads on EM and F1 across all three datasets in the LongBench setting while using **30% fewer tokens** than the next most accurate method. Finally, we introduce success economy—the amortized token cost per correct answer, weighted by difficulty—and advocate for efficiency-aware evaluation as a standard practice for agentic QA.

1 Introduction

Large language models encode vast world knowledge, yet retrieval-augmented generation remains essential for grounding their outputs in verifiable, up-to-date evidence (Gao et al., 2024; Lewis et al., 2021). Single-shot retrieval is insufficient for multi-hop questions, as each hop introduces dependencies that semantic similarity alone cannot resolve (Trivedi et al., 2022; Press et al., 2023; Singh et al., 2025). Agentic retrieval addresses this by giving models autonomy to iteratively search, reason, and accumulate evidence. Recent work augments these agents with knowledge graphs for structured multi-hop traversal (Li et al., 2024; Shen et al., 2025). However, this capability introduces a fundamental tension: **as questions become more complex, token usage grows rapidly**, driven by repeated LLM calls and expanding context across steps. This creates a key bottleneck for deploying agentic systems in practice.

Existing graph-based approaches to multi-hop QA fall along two tracks to optimize accuracy. First, *retrieval-augmented* methods such as HippoRAG (Gutiérrez et al., 2025) and GeAR (Shen et al., 2025) build triple-based knowledge graphs for structured traversal. However, recent work shows that triple extraction is both expensive and unreliable and often degrade retrieval quality below naive RAG (Zhuang et al., 2025; Han et al., 2026). We build upon this work and further show in §A.1.3 that KG triples are sub-optimal retrieval units, motivating our exploration of an alternative indexing strategy. Second, *extended-context* methods such

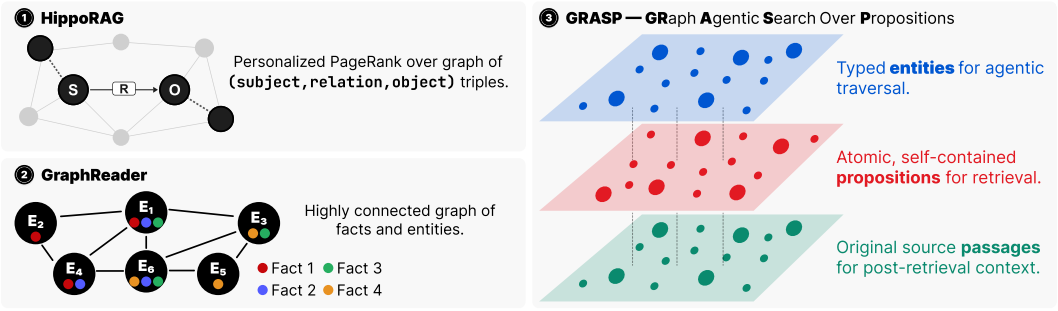

Figure 1: A comparison of graph structures for multi-hop QA. ❶ **Relational knowledge graphs**: HippoRAG extracts `(subject, relation, object)` triples into a flat knowledge graph and retrieves passages via Personalized PageRank. ❷ **Factual graphs**: GraphReader organizes atomic facts under key-element nodes into a highly connected graph, deployed by N parallel agents for traversal. ❸ **Proposition graph (ours)**: GRASP builds a hierarchical graph with three layers: typed entities for agentic traversal, propositions for retrieval, and source passages for post-retrieval context.

as GraphReader (Li et al., 2024) organize long texts into navigable graphs explored by multiple parallel agents that process redundant information. Across both tracks, no existing work optimizes for total token usage across both indexing and inference in the agentic setting.

In this paper, we propose GRASP: **Gr**aph **A**gentic **S**earch over **P**ropositions, an agentic retrieval system that jointly optimizes accuracy and token efficiency for multi-hop question answering. Rather than executing a singular retrieval query, GRASP decomposes multi-hop questions into dependency-aware plans, scaling the number of sub-agents to question complexity. Each sub-agent independently explores a three-layer hierarchical graph of entities, propositions, and passages constructed via joint extraction that avoids relational triples. Shared entities provide targeted gateways for traversal, propositions are a semantically rich retrieval unit for decomposed sub-questions, and passages supply full context for answering generation. Within each sub-agent, a compact observation state replaces growing message histories, bounding context growth over the life cycle of the agent.

In summary, our contributions are as follows:

1. We propose GRASP, a graph-enhanced agentic retrieval system that optimizes for both token efficiency and accuracy. We evaluate on MuSiQue, 2Wiki, and HotpotQA across the retrieval and extended context setting, reporting token usage for all methods. To our knowledge, GRASP is the first system evaluated competitively across both settings.
2. We introduce a novel three-layer hierarchical graph using propositions as the core retrieval unit, which achieves higher passage recall than HippoRAG2's triple-based graph at 69% fewer indexing tokens.
3. We design a planning-based inference pipeline with isolated sub-agents that compartmentalize context both across hops and within each sub-agent's execution.
4. We introduce *success economy*: a metric inspired by information theory that amortizes token cost over difficulty-weighted correct answers. Furthermore, we encourage the community to adopt efficiency-aware evaluation for agentic QA systems.

2 Related Work

Graph structures for retrieval. Graph-based retrieval captures inter-document relationships that flat indexing misses. HippoRAG (Gutiérrez et al., 2024) and HippoRAG2 (Gutiérrez et al., 2025) extract relational triples via OpenIE and retrieve passages through personalized PageRank (Figure 1). GeAR (Shen et al., 2025) adds LLM-guided beam search over a similar triple graph. LightRAG (Guo et al., 2025) proposes dual-level entity-and-relationship

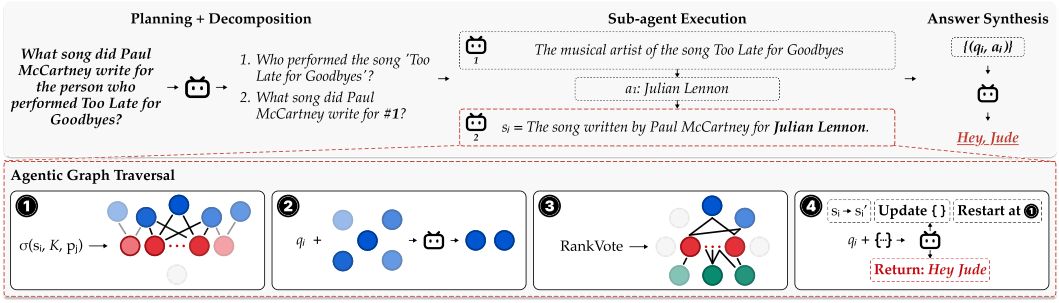

Figure 2: GRASP inference pipeline, illustrated with a MuSiQue example. **Top**: the planner decomposes the input into dependency-ordered sub-questions. A synthesis module produces the final answer from all sub-question–answer pairs. **Bottom**: agentic graph traversal for a single sub-agent. ❶ Retrieve propositions via hybrid search and aggregate to connecting entities. ❷ Select relevant entities. ❸ Score passage evidence by reciprocal rank vote. ❹ Return answer or rewrite the search statement and restart at ❶ (up to i iterations). Entity, proposition, and passage nodes are colored blue, red, and green, respectively.

indexing, though it still relies on relation extraction. However, recent work (Zhuang et al., 2025; Han et al., 2026) demonstrates that triple extraction is lossy—relations are context-dependent and difficult to compress into atomic triples without losing semantic nuance. GraphRAG (Edge et al., 2025) and RAPTOR (Sarthi et al., 2024) take summarization-based approaches, building hierarchical indices via community detection or recursive abstraction.

Agentic question answering. Yao et al. (2023); Park et al. (2023) established the paradigm of interleaving reflection, tool use, and memory in LLM agents. IRCoT (Trivedi et al., 2023) interleaves chain-of-thought with retrieval; Self-Ask (Press et al., 2023) decomposes questions into sequential sub-questions, a strategy we adopt in our planning module. Self-RAG (Asai et al., 2023) and Corrective RAG (Yan et al., 2024) introduce judges that control when to retrieve or re-retrieve, while FLARE (Jiang et al., 2023) uses generation confidence to trigger retrieval. Several methods couple agents with knowledge graph traversal: Think-on-Graph (Sun et al., 2024) and Reasoning on Graphs (Luo et al., 2024) prompt LLMs to explore existing KGs, while KGP (Wang et al., 2023) constructs a graph from document collections. In the extended-context setting, GraphReader (Li et al., 2024) links key elements through shared atomic facts (Figure 1) and deploys N parallel agents, but duplicates facts across nodes and lacks inter-agent coordination, resulting in redundant traversal and wasteful token consumption. ReadAgent (Lee et al., 2024) uses a gist memory index for selective recall without graph structure.

Proposition-based representations. FActScore (Min et al., 2023) introduced atomic fact decomposition for evaluating factual precision. PropSegmEnt (Chen et al., 2023a) provides a corpus for proposition segmentation and entailment, WiCE (Kamoi et al., 2023) extends claim-level entailment to Wikipedia verification, and Chen et al. (2023b) train sub-sentence encoders for proposition-level embeddings. Chen et al. (2024) adapted propositions to retrieval, showing that proposition-level indexing outperforms passage and sentence granularity for open-domain QA.

3 Method

GRASP is built on two core observations: (1) propositions, as atomic, self-contained natural language statements, are a more semantically faithful unit of retrieval than relational triples (§ 5, §A.1.3), and (2) joint proposition-entity extraction (Figure 3) produces a graph that is both computationally cheaper and easier to construct than triple-based alternatives. As illustrated in Figure 2, GRASP decomposes a complex question into dependency-ordered sub-questions, each resolved by a sub-agent that traverses the graph with iterative state updates, before a synthesis module produces the final answer.

3.1 Graph Construction

GRASP represents a corpus as a three-layer graph $\mathcal{G} = (\mathcal{V}_E, \mathcal{V}_P, \mathcal{V}_D, \mathcal{E})$, where $\mathcal{V}_E$, $\mathcal{V}_P$, and $\mathcal{V}_D$ denote entity, proposition, and passage nodes, respectively. Edges connect entities to the propositions from which they were extracted, and propositions to their source passages. We visualize GRASP's hierarchical structure in Figure 1.

Propositions and entities are extracted jointly in a single LLM call (Figure 3). We prompt the LLM to decompose the input into atomic, self-contained propositions and simultaneously identify named entities with typed labels, linking each entity to its source propositions via index references. These references directly form the entity–proposition edges in $\mathcal{G}$. The prompt operates over arbitrary input length, and each extracted proposition retains a link to its source passage. Entity nodes are deduplicated by canonical name and semantic type similarity, preventing graph fragmentation from minor type variation across passages. All propositions are embedded into vector representations by a pre-trained language encoder for hybrid search at inference time. We show in §5 that proposition extraction provides a significant boost in recall over raw sentences, justifying the extraction cost.

3.2 Agent Planning

Given a multi-hop question, a planner module produces a depdency-aware plan and an ordered sequence of sub-questions. The plan is a brief natural language outline of the reasoning chain, while the sub-questions decompose the original question into single-hop retrieval tasks, each targeting a specific fact. Furthermore, we adapt the output format to model inter-question dependencies explicitly. Sub-questions reference the answers of prior sub-questions through numeric placeholders (e.g., *Where was #1 born?*), similar to the dependency structure of MuSiQue. The planner is implemented as a single LLM call with few-shot examples covering linear, branching, and comparison questions (Figure 7).

3.3 Sub-Agent Execution

The bottom half of Figure 2 visualizes the life cycle of each sub-agent. Specifically, each sub-question q_i from the plan is resolved by an independent sub-agent α_i that is conditioned on three inputs: the overarching multi-hop question Q, the sub-question q_i itself, and the history of preceding question-answer pairs $\mathcal{H} = \{(q_1, a_1), \ldots, (q_{i-1}, a_{i-1})\}$. In a query rewriting step, the sub-agent leverages these inputs to resolve dependencies in q_i and to reformulate it into a declarative search statement s_i that is used for dense semantic search over the graph's propositions. To ensure exact term matching alongside semantic similarity, the sub-agent simultaneously generates a keyword list $\mathcal{K}$ for the lexical component of the hybrid search function in Eq. (1). The goal of retrieval at this stage is to return a relevant set of candidate entities $\mathcal{C} \subset \mathcal{V}_E$ from the graph for the agent to begin traversal. It begins with a hybrid search (dense vector search + keyword search) over $\mathcal{V}_P$ to retrieve a subset $\mathcal{P}_m$ consisting of the top m propositions. These propositions are scored by a weighted sum of semantic similarity and keyword-based BM25:

$$\sigma(s_i, \mathcal{K}, p_j) = \cos(\mathbf{e}_{s_i} \mathbf{e}_{p_j}) + \lambda \cdot \log\big(1 + \mathrm{BM25}(\mathcal{K}, p_j)\big), \tag{1}$$

where $\mathbf{e}_s$ and $\mathbf{e}_{p_j}$ are the dense embeddings of the search statement s_i and proposition p_j, respectively. These scores are aggregated to linked entity nodes:

$$\mathrm{score}(e_i) = \frac{\sum_{p_j \in \mathcal{P}_m(e_i)} \sigma(s_i, \mathcal{K}, p_j)}{\sqrt{1 + d(e_i)}}, \tag{2}$$

where $\mathcal{P}_m(e_i)$ denotes the propositions linked to e_i and $d(e_i)$ is its degree. This normalization dampens hub entities that appear in many propositions. The sub-agent then compares the candidates $\mathcal{C} \subset \mathcal{V}_E$ against q_i, selecting a subset $\mathcal{E} \subseteq \mathcal{C}$ for traversal. This filtering step is necessary to give the sub-agent autonomy in directing the traversal and to restrict the number of propositions involved in later voting for passage-level retrieval (see §A.1.7 for empirical justification).

INPUT PASSAGE

"The earliest evidence for the Easter Hare (Osterhase) was recorded in south-west Germany in 1678 by the professor of medicine Georg Franck von Franckenau, but it remained unknown in other parts of Germany until the 18th century."

PROPOSITIONS

```
[0] "The earliest evidence for the Easter Hare was recorded in south-west Germany in 1678 by Georg Franck von Franckenau."
[1] "Georg Franck von Franckenau was a professor of medicine."
[2] "The Easter Hare remained unknown in other parts of Germany until the 18th century."
[3] "The Easter Hare is also known as Osterhase."
```

ENTITIES

NAME	TYPE	PROP. INDICES
Easter Hare	Folklore Figure	[0, 2, 3]
George Frank Von Franck von Franckenau	Professor of Medicine	[0, 1]
Germany	Country	[0, 2]
1678	Year	[0]
Osterhase	Folklore Figure	[3]

Figure 3: Proposition and entity extraction from a single passage. The passage is decomposed into propositions, with entities extracted simultaneously and linked to propositions via index references. These links form the entity–proposition edges of the graph (§3.1).

As shown in step ❸ of Figure 2, sub-agent α_i retrieves passage-level evidence by using s_i to perform a semantic search over all propositions linked to entities in $\mathcal{E}$, i.e., $\bigcup_{e_i \in \mathcal{E}} \mathcal{P}(e_i)$. Passages are scored using the `RankVote` (Geirhos et al., 2025) of their constituent propositions. Because each proposition p_j belongs to exactly one parent passage D, a passage's total score is computed by summing the reciprocal-rank votes of its connected propositions: $\text{score}(D) = \sum_{p_j \in \mathcal{P}(D)} w_j$, where $\mathcal{P}(D)$ is the set of propositions extracted from D, $w_j = \frac{1}{1+R(p_j)}$ is the vote weight, and $R(p_j)$ is the proposition's rank. This scoring function ranks the passages in $\mathcal{V}_D$ and returns the top d to the sub-agent as new evidence. We ablate `RankVote` against uniform weighting in § 5.

Instead of tracking a continuous message history across its entire life cycle, GRASP equips each sub-agent with a compact observation state O_i. At each LLM call, α_i processes its state alongside freshly retrieved evidence to generate concise, entity-level observations that are appended back to O_i. Furthermore, α_i filters previously visited entities and propositions from subsequent retrieval, ensuring α_i does not revisit evidence. This design bounds context growth at two levels. Within a sub-agent, O_i replaces a full conversation history; each entry is a concise summary, and previously retrieved evidence is never re-read. Across sub-agents, only Q and $\mathcal{H}$ propagate—discarding all tool calls, retrieved evidence, and intermediate reasoning traces.

After observing both O_i and new evidence, α_i selects one of two actions (step ❹, Figure 2):

- **DONE**: evidence is sufficient; return answer a_i.
- **QUERY_AGAIN**: evidence is insufficient; rewrite the search statement $s_i \rightarrow s_i'$, add necessary observations to $_i$, and restart traversal at step ❶.

After all sub-agents complete, $\mathcal{H}$ is passed to a synthesis module that produces the final answer. This module consolidates the sub-answers into a single coherent response. In practice, this step primarily serves to format final answers for evaluation against ground-truth annotations.

4 Experimental Setup

Datasets. We evaluate on three widely used multi-hop reasoning datasets: MuSiQue (Trivedi et al., 2022), 2WikiMultihopQA (Ho et al., 2020), and HotpotQA (Yang et al., 2018). These benchmarks have been adopted by recent graph-based and agentic QA systems, including HippoRAG, HippoRAG2, IRCoT, GeAR, GraphReader, and LinearRAG.

Each dataset originally provides per-question passage sets containing both gold supporting passages and distractors. We evaluate in two settings that repurpose these datasets differently. In the *retrieval setting*, passages are pooled into a shared index and systems must retrieve relevant context at inference time. We use the HippoRAG2 splits, each containing

1,000 question-passage-set pairs with ground-truth passage labels and final answers. In the *extended context setting*, we use the LongBench (Bai et al., 2024) variants of the same datasets, where ground-truth passages are distributed among distractors in a single long input. Each LongBench split contains 200 samples with varying context lengths. Full statistics are provided in Table 7.

Baselines. We select baselines that incorporate some level of LLM autonomy over the retrieval or reasoning process and construct an index over the corpus before inference, enabling a fair comparison of agentic architectures. Non-agentic, non-iterative baselines are reported in the appendix (Table 5), as well as baselines we considered but ultimately excluded. All methods use the same `gemini` LLM backbone for indexing and inference, as well as the same `gemini-embedding-001` encoder to extract representations for retrieval. Further implementation details are found in section A.1.5.

In the *retrieval setting*, we compare against IRCoT+HippoRAG2 (Trivedi et al., 2023; Gutiérrez et al., 2025), which interleaves chain-of-thought retrieval with a triple-based graph retriever, and GeAR (Shen et al., 2025), which uses LLM-guided beam search over a triple graph. In the *extended context setting*, we use GraphReader (Li et al., 2024) with N=5 parallel agents, a single-agent variant GraphReader (N=1) to isolate the effect of parallelism, and ReadAgent (Lee et al., 2024), which uses gist memory for selecting passages.

Evaluation metrics. We evaluate answer quality with Exact Match (EM), token-level F1, and LLM-as-a-Judge, following standard practice in multi-hop QA evaluation. We use the judge prompts from ReadAgent (and later in GraphReader). To quantify efficiency, let $\mathcal{T}_i$ denote the total token usage for question Q_i, aggregated across all LLM calls during both indexing and inference. We report two efficiency metrics:

Total token usage. The sum $\sum_{i=1}^{N} \mathcal{T}_i$ across all questions, capturing end-to-end cost.

Success economy. To capture token efficiency, we amortize token usage over correct answers, weighted by difficulty. Classical information theory provides a mechanism via *surprisal* (Shannon, 1948) : the self-information $w_i = -\log_2(r_i)$ quantifies the bits of information gained from observing a correct answer to q_i with probability r_i. We estimate r_i by prompting the base LLM f with q_i n-times at non-zero temperature and computing the fraction of correct answers. This naturally assigns higher weight to questions f rarely solves and lower weight to those it handles trivially, following the principle that not all evaluation items are equally informative (Rodriguez et al., 2021). We formally define success economy as:

$$C_w = \frac{\sum_{i=1}^{N} \mathcal{T}_i}{\sum_{i=1}^{N} w_i \cdot \mathbb{1}[\mathrm{EM}_i = 1]} \tag{3}$$

Lower is better. This penalizes token waste, low accuracy, and easy-question accumulation.

5 Experiments

Question answering evaluation. Table 1 reports our main results across both evaluation settings.

Retrieval setting. GRASP achieves the highest EM, F1, and Judge scores on MuSiQue (0.505, 0.651, 0.765) and 2Wiki (0.737, 0.825, 0.943), outperforming IRCoT+HippoRAG2 across all metrics while using roughly **50% fewer total tokens**. On HotpotQA, IRCoT+HippoRAG2 leads on accuracy, but at more than 2× the token cost (26.82M vs. 12.68M). Though, in section A.1.4 we explain why Hotpot in the retrieval setting is the least informative dataset in our experiments. GeAR beats out IRCoT+HippoRAG2 on MuSiQue for Judge, but trails behind all other baselines in 2Wiki and HotpotQA on EM, F1 and Judge scores. **GRASP scores the best success economy on all three datasets**, indicating that its token budget is more effectively utilized toward correct answers.

Extended context setting. **GRASP leads on EM and F1 across all three datasets while using substantially fewer tokens than GraphReader**. On MuSiQue, GraphReader edges GRASP on Judge by 0.005 (0.730 vs. 0.725)—a difference of a single question—while using 60% more tokens. The gap is widest on 2Wiki, where GRASP improves EM by 12 points over

Dataset	Setting	Method	Accuracy			Efficiency	
			EM	F1	Judge	Tokens (M)	$C_w \downarrow$
MuSiQue	Retrieval	GRASP (Ours)	**0.505**	**0.651**	**0.765**	**15.89**	**12,616**
		IRCoT + HippoRAG2	0.478	0.604	0.673	31.49	26,655
		GeAR	0.414	0.530	0.689	16.69	15,545
	Extended Ctx.	GRASP (Ours)	**0.510**	**0.643**	0.725	**8.87**	**46,707**
		GraphReader	0.480	0.619	**0.73**	14.07	80,852
		GraphReader (N=1)	0.400	0.538	0.63	9.44	73,719
		ReadAgent	0.355	0.466	0.555	14.07	116,670
2Wiki	Retrieval	GRASP (Ours)	**0.737**	**0.825**	**0.943**	10.9	**9,477**
		IRCoT + HippoRAG2	0.717	0.805	0.913	21.52	20,245
		GeAR	0.444	0.478	0.542	**9.89**	13,718
	Extended Ctx.	GRASP (Ours)	**0.710**	**0.810**	**0.935**	6.18	**27,378**
		GraphReader	0.590	0.764	**0.935**	8.84	53,298
		GraphReader (N=1)	0.580	0.741	0.875	**4.96**	32,917
		ReadAgent	0.520	0.654	0.820	6.08	39,268
HotpotQA	Retrieval	GRASP (Ours)	0.627	0.772	0.898	**12.68**	**8,847**
		IRCoT + HippoRAG2	**0.678**	**0.807**	**0.924**	26.82	26,370
		GeAR	0.557	0.677	0.793	13.18	16,450
	Extended Ctx.	GRASP (Ours)	**0.540**	**0.704**	**0.890**	**5.43**	**47,668**
		GraphReader	0.515	0.679	0.870	12.23	76,367
		GraphReader (N=1)	0.485	0.640	0.825	7.96	56,417
		ReadAgent	0.445	0.612	0.820	11.53	99,831

Table 1: Results in the retrieval and extended context settings on MuSiQue, 2Wiki, and HotpotQA. Best results per setting are **bolded**. Tokens in millions (M). C_w: difficulty-weighted success economy (tokens per weighted correct answer; $\downarrow$: lower is better).

GraphReader (0.710 vs. 0.590) and matches its Judge score (0.935) at roughly **70% of the token cost**. On HotpotQA, GRASP achieves a 0.890 Judge compared to GraphReader's 0.870 while using less than half the tokens (5.43M vs. 12.23M). GraphReader (N=1) reduces tokens by 20–40% but suffers accuracy drops of up to 11% EM and 10% Judge, suggesting that naive cost reduction through removing GraphReader's parallelism noticeably degrades the agent's QA performance. ReadAgent is the only baseline without graph structure and consistently trails on all metrics, confirming that graph-based exploration provides a meaningful advantage for multi-hop reasoning. **GRASP again has the lowest success economy on all datasets**.

Retrieval evaluation. We benchmark the effectiveness of GRASP's retrieval mechanism against HippoRAG2 under two configurations on the MuSiQue dataset, evaluating passage-level recall for both:

Simulated agentic retrieval. To simulate the sub-agent retrieval process during QA in GRASP, we utilize MuSiQue's annotations to extract sub-questions. For each sub-question, GRASP retrieves the top $k = 5$ passages. This setting measures the upper-bound retrieval performance of GRASP's decomposed approach. Conversely, HippoRAG2 performs a *single retrieval pass* using the entire multi-hop question. To establish the fairest comparison possible and ensure both methods retrieve the exact same total number of passages, we dynamically extend HippoRAG2's retrieval limit to $k_i = 5 \times n^{(i)}_{\textbf{hops}}$ for a given question i.

Standard single-pass retrieval. To provide a direct and strict comparison to HippoRAG2's default evaluation setting, we restrict the total retrieved passages to $k = 5$ to both methods. GRASP abandons sub-question decomposition and instead uses the original, multi-hop question Q to query its graph in a single pass.

Retrieval results. The results in Table 2 demonstrate the effectiveness GRASP in retrieving ground truth passages. Under the simulated agentic setting, the full GRASP configuration achieves a recall of **0.904**, outperforming HippoRAG2's 0.817. While GRASP's retrieval

Mode	Method	Retrieval Unit	Weight	Recall	Idx. Tokens
Sim. Agent $k_i = 5 \times n^{(i)}_{\textbf{hops}}$	HippoRAG2	–	–	0.818	17.6M
	GRASP*	Proposition	RankVoting	**0.904**	5.4M
	GRASP	Proposition	Uniform	0.848	
	GRASP	Sentence	RankVoting	0.853	3.3M
	GRASP	Sentence	Uniform	0.648	
Single-Pass $k = 5$	HippoRAG2	–	–	0.702	17.6M
	GRASP	Proposition	RankVoting	**0.729**	5.4M
	GRASP	Proposition	Uniform	0.610	
	GRASP	Sentence	RankVoting	0.656	3.3M
	GRASP	Sentence	Uniform	0.429	

*Configuration used in the GRASP QA experiments.

Table 2: Retrieval performance on the MuSiQue. Split from Gutiérrez et al. (2025).

mechanism was not designed for multi-hop retrieval in a single-pass, it surprisingly maintains an advantage even in this setting, achieving 0.729 compared to HippoRAG2's 0.702.

Propositions as retrieval units. Using extracted propositions rather than raw sentences improves recall (e.g., from 0.853 to 0.904 under RankVoting weighting), demonstrating they are a superior retrieval unit. While indexing propositions naturally incurs a higher token cost than sentences (5.4M vs. 3.3M), this trade-off is favorable; GRASP still requires 69% fewer indexing tokens than HippoRAG2 (17.6M).

RankVoting. To aggregate retrieval unit hits to passages, we use RankVoting as described in Geirhos et al. (2025), which outperforms uniform weighting. RankVoting allots each retrieval unit a weighted-vote by its reciprocal rank in the set. By weighting passages according to the ranks of their matched propositions, GRASP effectively prioritizes highly relevant context, yielding a nearly 6-point recall improvement over uniform weighting in the simulated agentic setting and 11-point increase in the single-pass setting.

Graph Structure vs. Inference Architecture. To disentangle the contributions of GRASP's proposition graph from its inference architecture, we cross each graph structure with each traversal method on MuSiQue (Table 3).

		Accuracy			Efficiency	
Graph	**Agent**	EM	F1	Judge	Tokens (M)	$C_w \downarrow$
Retrieval setting						
Proposition	GRASP	**0.505**	**0.651**	**0.765**	**15.89**	**12,616**
HippoRAG triple	GRASP	0.496	0.638	0.756	29.12	23,786
HippoRAG triple	IRCoT	0.478	0.604	0.673	31.49	26,655
Long-context setting						
Proposition	GRASP	**0.510**	**0.643**	0.725	**8.87**	**46,707**
Proposition	GraphReader	0.480	0.627	0.722	13.82	80,800
Key-element	GraphReader	0.480	0.619	**0.730**	14.07	80,852
Key-element	GRASP	0.495	0.606	0.690	18.16	93,716

Table 3: Cross-ablation of graph structure and inference architecture on MuSiQue. Shaded rows are original configurations from Table 1; unshaded rows swap the graph or traversal method. Tokens in millions (M). ↓: lower is better.

Retrieval setting. Running GRASP's agent over HippoRAG2's triple graph yields comparable accuracy but nearly doubles success economy due to expensive triple extraction. Notably, this variant still improves over IRCoT+HippoRAG2 in both accuracy and success economy ,

suggesting that decomposition-based planning is more effective than interleaving retrieval with chain-of-thought, independent of the underlying graph.

Extended context setting. GraphReader over GRASP's proposition graph matches the original GraphReader in accuracy and success economy. This is not surprising since GraphReader traverses via key-element expansion without embedding-based retrieval, so it does not utilize the proposition graph's dense embeddings. Conversely, GRASP's agent over GraphReader's key-element graph produces the worst success economy of any configuration. Without embedded nodes, the agent cannot perform hybrid search and must rely entirely on LLM-driven traversal, inflating token usage without a corresponding accuracy gain. This contrast highlights that LLM-driven traversal alone is expensive and insufficient. Pairing it with embedding-based retrieval at the proposition layer is what enables GRASP's favorable accuracy-to-cost tradeoff.

Planner Analysis. Table 4 evaluates the planner's decomposition quality on MuSiQue by comparing the number of planned steps to the ground-truth hop count. Overall, the planner produces the correct number of steps for 82.2% of questions, though accuracy degrades from 89.4% at 2-hop to 66.9% at 4-hop. When the planner matches the true hop count, EM improves by nearly 9 points (52.1% vs. 43.3%), confirming that planning and decomposition quality directly impacts downstream QA accuracy.

This result also helps explain a pattern in Table 3: despite the proposition graph achieving substantially higher retrieval recall than HippoRAG2's triple graph (0.904 vs. 0.818, Table 2), the end-to-end EM gap is only 1 point (0.505 vs. 0.496). Planning errors impose a ceiling that limits how much better retrieval can translate into better answers. When the decomposition is wrong, errors propagate through the sub-agent chain regardless of graph quality. GRASP currently generates plans via few-shot examples in context. Improving planner accuracy via supervised fine-tuning or reinforcement learning with verifiable rewards, particularly for higher-hop questions, is a promising direction for future work.

Hops	N	Plan Acc.	Avg Dev.	EM (match)	EM (no match)
2	518	89.4%	+0.08	57.2%	47.3%
3	316	78.5%	-0.06	50.0%	48.5%
4	166	66.9%	-0.36	35.1%	32.7%
Overall	1000	82.2%	-0.04	52.1%	43.3%

Table 4: Planner hop accuracy on MuSiQue. Plan Acc. is the fraction of questions where the number of planned steps equals the true hop count. Avg Dev. is the mean signed deviation (planned $-$ true). EM (match) and EM (no match) are exact-match scores split by whether the planner generated the correct number of steps.

6 Conclusion

We presented GRASP, a graph-enhanced agentic retrieval system that treats token efficiency as a first-class design objective alongside QA accuracy for multi-hop question answering. By constructing a three-layer hierarchical graph from jointly extracted propositions and entities, GRASP avoids the cost and noise of relational triple extraction while producing a richer retrieval unit. Its planning-based inference pipeline decomposes multi-hop questions into single-hop sub-tasks handled by sub-agents, compartmentalizing context to keep token usage controlled as reasoning depth increases.

Experiments on MuSiQue, 2WikiMultihopQA, and HotpotQA show that GRASP achieves the best accuracy-to-cost tradeoff across both the retrieval and long-context settings, sitting on the Pareto frontier in nearly every configuration. We introduce *success economy*, a metric inspired by information theory that amortizes token cost over difficulty-weighted correct answers. We hope that this encourages the community to adopt efficiency as a standard evaluation dimension for agentic QA systems, where inference cost scales directly with reasoning complexity.

Acknowledgment

This research was supported in part by the National Science Foundation under Award #IIS-2449768 and the Technical AI Safety Research Program at Coefficient Giving. The views and conclusions expressed in this work are those of the authors and should not be interpreted as representing the official policies or endorsements of the U.S. Government, the National Science Foundation, or Coefficient Giving.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023. URL https://arxiv.org/abs/2310.11511.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding, 2024. URL https://arxiv.org/abs/2308.14508.

Jifan Chen and Greg Durrett. Understanding dataset design choices for multi-hop reasoning, 2019. URL https://arxiv.org/abs/1904.12106.

Sihao Chen, Senaka Buthpitiya, Alex Fabrikant, Dan Roth, and Tal Schuster. PropSegmEnt: A large-scale corpus for proposition-level segmentation and entailment recognition. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8874–8893, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.565. URL https://aclanthology.org/2023.findings-acl.565/.

Sihao Chen, Hongming Zhang, Tong Chen, Ben Zhou, Wenhao Yu, Dian Yu, Baolin Peng, Hongwei Wang, Dan Roth, and Dong Yu. Sub-sentence encoder: Contrastive learning of propositional semantic representations, 2023b. URL https://arxiv.org/abs/2311.04335.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. Dense x retrieval: What retrieval granularity should we use?, 2024. URL https://arxiv.org/abs/2312.06648.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2025. URL https://arxiv.org/abs/2404.16130.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

Robert Geirhos, Priyank Jaini, Austin Stone, Sourabh Medapati, Xi Yi, George Toderici, Abhijit Ogale, and Jonathon Shlens. Towards flexible perception with visual memory, 2025. URL https://arxiv.org/abs/2408.08172.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2410.05779.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=LWH8yn4HS2.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models, 2024. URL https://arxiv.org/abs/2405.14831.

Haoyu Han, Li Ma, Yu Wang, Harry Shomer, Yongjia Lei, Zhisheng Qi, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, Charu C. Aggarwal, and Jiliang Tang. Rag vs. graphrag: A systematic evaluation and key insights, 2026. URL https://arxiv.org/abs/2502.11371.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, 2020. URL https://arxiv.org/abs/2011.01060.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation, 2023. URL https://arxiv.org/abs/2305.06983.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering, 2019. URL https://arxiv.org/abs/1909.06146.

Ryo Kamoi, Tanya Goyal, Juan Diego Rodriguez, and Greg Durrett. Wice: Real-world entailment for claims in wikipedia, 2023. URL https://arxiv.org/abs/2303.01432.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts, 2024. URL https://arxiv.org/abs/2402.09727.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL https://arxiv.org/abs/2005.11401.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, Wenbo Su, and Bo Zheng. Graphreader: Building graph-based agent to enhance long-context abilities of large language models, 2024. URL https://arxiv.org/abs/2406.14550.

Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, Huaidong Xiong, Lin Yuan, Jun Xu, Zaoyang Wang, Zhiqiang Zhang, Wen Zhang, Huajun Chen, Wenguang Chen, and Jun Zhou. Kag: Boosting llms in professional domains via knowledge augmented generation, 2024. URL https://arxiv.org/abs/2409.13731.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning, 2024. URL https://arxiv.org/abs/2310.01061.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. Compositional questions do not necessitate multi-hop reasoning, 2019. URL https://arxiv.org/abs/1906.02900.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation, 2023. URL https://arxiv.org/abs/2305.14251.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023. URL https://arxiv.org/abs/2304.03442.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023. URL https://arxiv.org/abs/2210.03350.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016. URL https://arxiv.org/abs/1606.05250.

Pedro Rodriguez, Joe Barrow, Alexander Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. Evaluation examples are not equally informative: How should that change NLP leaderboards? In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *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)*, pp. 4486–4503, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.346. URL https://aclanthology.org/2021.acl-long.346/.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. Raptor: Recursive abstractive processing for tree-organized retrieval, 2024. URL https://arxiv.org/abs/2401.18059.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

Zhili Shen, Chenxin Diao, Pavlos Vougiouklis, Pascual Merita, Shriram Piramanayagam, Enting Chen, Damien Graux, Andre Melo, Ruofei Lai, Zeren Jiang, Zhongyang Li, YE QI, Yang Ren, Dandan Tu, and Jeff Z. Pan. Gear: Graph-enhanced agent for retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2412.18431.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag, 2025. URL https://arxiv.org/abs/2501.09136.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph, 2024. URL https://arxiv.org/abs/2307.07697.

Chihiro Taguchi, Seiji Maekawa, and Nikita Bhutani. Efficient context selection for long-context qa: No tuning, no iteration, just adaptive-k, 2025. URL https://arxiv.org/abs/2506.08479.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition, 2022. URL https://arxiv.org/abs/2108.00573.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions, 2023. URL https://arxiv.org/abs/2212.10509.

Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. Knowledge graph prompting for multi-document question answering, 2023. URL https://arxiv.org/abs/2308.11730.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation, 2024. URL https://arxiv.org/abs/2401.15884.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018. URL https://arxiv.org/abs/1809.09600.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. Linearrag: Linear graph retrieval augmented generation on large-scale corpora, 2025. URL https://arxiv.org/abs/2510.10114.

A Appendix

Summary

This appendix supplements the main paper with non-agentic baseline comparisons, a discussion of HotpotQA data integrity issues, implementation details, success economy stratified by hop count, and structural ablations validating each layer of GRASP's index. Also, all prompts used in GRASP's agent framework are provided in Section A.2.

A.1 Experiments

A.1.1 Non-agentic Baselines

To quantify the value of agentic retrieval, we measure accuracy for four non-iterative baselines: **No Context** (zero-shot Gemini), **RAG** (single-pass retrieval with Gemini), **Self-Ask + RAG** (decomposition-based prompting from Press et al. (2023) with single-pass dense retrieval per sub-question), and **Full Context** (Gemini given the complete LongBench passage set).

As shown in Table 5, GRASP consistently outperforms these baselines across all metrics. In the retrieval setting, GRASP surpasses single-pass RAG by 13.3, 25.4, and 2.4 EM points on MuSiQue, 2Wiki, and HotpotQA, respectively, with uniform gains in F1 and LLM Judge scores. Self-Ask + RAG provides a useful intermediate comparison: it improves over flat RAG on 2Wiki (+17 EM), where sub-questions are cleanly separable compositional lookups, but under performs flat RAG on MuSiQue (-3 EM). Without structured retrieval to ground each sub-question's evidence, errors in decomposition cascade through the chain. GRASP outperforms both baselines on all three datasets, indicating that decomposition must be paired with structured retrieval to benefit consistently.

Furthermore, in the extended context setting, GRASP substantially outperforms the Full Context baseline, where Gemini is given all passages simultaneously, by 15 to 34 EM points. This demonstrates that structured, agentic graph exploration extracts and synthesizes relevant evidence far more effectively than simply expanding an LLM's context window for multi-hop reasoning problems.

		MuSiQue			2Wiki			HotpotQA		
Setting	**Method**	EM	F1	Judge	EM	F1	Judge	EM	F1	Judge
Retrieval	No Context	.178	.291	.359	.396	.490	.575	.395	.532	.677
	RAG	.372	.466	.551	.483	.570	.685	.603	.733	.876
	Self-Ask + RAG	.332	.438	.530	.653	.732	.829	.575	.722	.866
	GRASP (Ours)	**.505**	**.650**	**.765**	**.737**	**.825**	**.943**	**.627**	**.772**	**.898**
Extended Context	Full Context	.360	.449	.540	.370	.545	.785	.385	.566	.805
	GRASP (Ours)	**.510**	**.643**	**.725**	**.710**	**.810**	**.935**	**.540**	**.704**	**.890**

Table 5: Comparison with non-agentic baselines. No Context: zero-shot Gemini. RAG: single-pass retrieval. Self-Ask + RAG: decomposition-based prompting (Press et al., 2023) with single-pass dense retrieval per sub-question. Full Context: all LongBench passages provided.

A.1.2 GRASP Error Analysis

To diagnose where GRASP could be improved, we manually inspect the trace of 50 incorrect predictions from MuSiQue (retrieval setting) to identify where residual errors originate. We also categorize each along two axes: by primary cause (*retrieval*, *reasoning*, or *dataset*) and by the pipeline stage at which the failure was introduced. Results are reported in table 6.

Our analysis shows that 6 cases (12%) are dataset-level failures in which the question targets one real-world entity but the gold answer requires a different entity of the same name

(e.g., Atlanta, GA vs. Atlanta, MI; Francis Bacon the painter vs. the philosopher). The agent correctly recognizes the two entities as distinct, but the evaluation penalizes it for refusing to bridge them. Of the remaining 44 agent errors, **36 are reasoning-side and only 8 are retrieval-side** — a 4.5:1 ratio that empirically validates our design choice in §3.1: the proposition graph achieves high passage recall (Table 2) and is rarely the bottleneck.

The 36 reasoning-side errors split between **planner errors** (13, almost entirely on 3- and 4-hop questions, consistent with the planner-accuracy decline in Table 4) and **sub-agent judgment errors** (23) in which the retrieved evidence and resolved context are sufficient to answer correctly but the sub-agent reaches a wrong conclusion. We observe four recurring causes: *salience bias* (picking the most prominent among several valid candidates in evidence), *scope or granularity mismatch* (answering at the wrong aggregation level), *parametric leakage* (overriding retrieved evidence with world knowledge), and *semantic misreading* (the sub-agent misinterprets the sub-question and extracts an adjacent-but-wrong fact). Both the planner and sub-agent in GRASP are currently realized through prompting. Learning these control policies, as in Self-RAG and Search-R1 (§2), is the natural lever for reducing the reasoning errors mentioned.

Category	Plan	Step 0	Step 1	Step 2	Step 3	N/A	Total
Retrieval Wrong Node	–	1	–	–	1	–	2
Retrieval Missing Fact	–	1	3	1	1	–	6
Reasoning Judgment Error	–	4	10	8	1	–	23
Reasoning Wrong Plan	13	–	–	–	–	–	13
Dataset Error	–	–	–	–	–	6	6
Total	13	6	13	9	3	6	50

Table 6: Manual error analysis on 50 incorrect predictions, stratified by the pipeline stage at which the failure was introduced. "Plan" denotes errors originating in the planner's decomposition; "Step k" denotes errors made by the k-th sub-agent during execution; "N/A" marks cases where the underlying dataset — not the agent — is at fault (e.g. same-name entity collisions across cities or people). Planner errors and sub-agent mis-extraction at the middle hops (Steps 1–2) account for the bulk of failures.

A.1.3 Retrieval Unit Embedding Discriminability

To understand why proposition-based retrieval outperforms KG triple–based methods, we investigate whether the choice of retrieval unit affects how well a query embedding can discriminate relevant units from irrelevant ones — independent of the retrieval pipeline itself. While prior work such as LinearRAG (Zhuang et al., 2025) has attributed KG failures to extraction errors (e.g., negation inversions), we hypothesize a more fundamental issue: even correctly extracted triples are inferior embedding targets, as compressing information into subject–predicate-object form discards contextual nuance that embedding models rely on to match natural-language queries.

To test this, we sample 1,000 questions from three single-hop retrieval benchmarks — SQuAD (Rajpurkar et al., 2016), Natural Questions (Kwiatkowski et al., 2019), and PubMedQA (Jin et al., 2019) — and extract retrieval units from each question's gold passages using three methods: proposition-based extraction (GRASP), triple-based extraction (HippoRAG and GeAR), and naive sentence splitting. Sentence splitting serves as a format-controlled baseline: like propositions it produces natural-language units, but without atomic decomposition. All units from all 1,000 questions are pooled into a shared index per method; for each question, every unit in the pool is ranked by cosine similarity to the query, with gold units treated as relevant. We report NDCG@5, measuring whether units from the correct passage are surfaced near the top of the full ranking.

As shown in Figure 4, propositions achieve the highest discriminability across all three datasets, outperforming the strongest triple-based baseline (HippoRAG) by 8.1, 4.6, and 10.0 points on SQuAD, Natural Questions, and PubMedQA respectively. The sentence baseline illuminates the structure of this advantage. Sentences substantially outperform GeAR across

all datasets (+7.6, +2.7, +8.9 points), and outperform HippoRAG on PubMedQA (+6.4 points) where dense biomedical phrasing resists clean triple decomposition — implicating triple compression itself, rather than extraction errors, as a primary driver of the gap. Propositions improve further upon sentences by 3.6–8.4 points across all three datasets, indicating that atomic decomposition into independently retrievable units provides discriminability benefits beyond natural-language format alone. Together, these results suggest that the embedding alignment advantage of propositions has two complementary sources: preserving natural-language form and decomposing passages into atomic, query-aligned units.

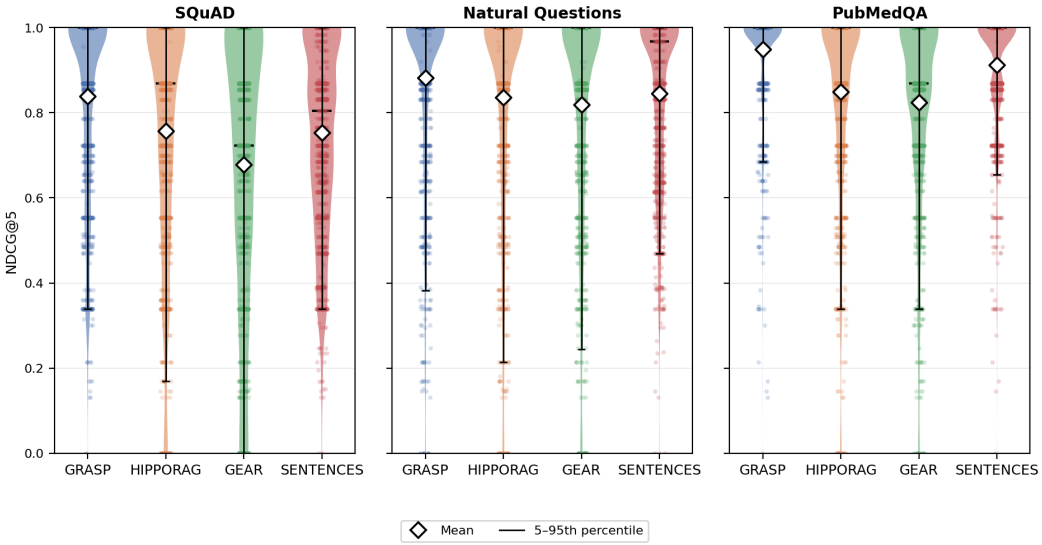

Figure 4: **NDCG@5 for retrieval unit discriminability**, measured by ranking all 1,000 questions' extracted units in a shared pool per method and treating gold units as relevant. Propositions (GRASP) achieve the highest discriminability across all three benchmarks. The sentence baseline — natural-language units without atomic decomposition — substantially outperforms GeAR across all datasets and HippoRAG on PubMedQA, implicating triple compression as the primary bottleneck. Propositions improve further upon sentences, showing that atomic decomposition provides an additional benefit.

A.1.4 Data Integrity Issues in HotpotQA

HotpotQA is increasingly viewed as a near-solved dataset, diminishing its utility as a rigorous benchmark for evaluating agentic multi-hop reasoning. As highlighted by the creators of MuSiQue (Trivedi et al., 2022) and prior studies (Min et al., 2019; Chen & Durrett, 2019), HotpotQA contains pervasive reasoning shortcuts that allow models to bypass genuine step-by-step deduction. Furthermore, because the dataset is widely circulated and based on general Wikipedia knowledge, its answers are deeply ingrained in the parametric memory of modern large language models. This allows models to "cheat" by relying on internalized facts rather than synthesizing evidence across retrieved passages. This parametric leakage largely explains our observation that zero-shot Gemini achieves a surprisingly competitive score on HotpotQA even in the **No Context** setting, whereas its closed-book performance sharply degrades on more rigorously constrained datasets like MuSiQue.

A.1.5 Implementation Details

All LLM calls use `gemini-3-flash-preview` with `thinking_level="minimal"` via the Google AI API, and all embeddings are generated using `gemini-embedding-001`. We use Neo4j for graph storage as an implementation convenience, not an architectural requirement. For model hyperparameters, we set the temperature to 0.2 for LLM generators to allow for slight flexibility in reasoning, and use a temperature of 0.1 for LLM extractors.

	Retrieval Setting			Extended context Setting		
	MuSiQue	2Wiki	HotpotQA	MuSiQue	2Wiki	HotpotQA
# queries	1,000	1,000	1,000	200	200	200
# passages	11,656	6,119	9,811	2,218	1,986	1,722
Avg. # tokens	2,344	903	1,300	15,976	13,148	7,227

Table 7: Dataset statistics for both evaluation settings. Retrieval splits from HippoRAG2 (Gutiérrez et al., 2025); Extended context splits from LongBench (Bai et al., 2024). Token count is derived from Gemini AI API for `gemini-3-flash-preview`, the LLM backbone used across all of our experiments.

During graph construction, we process 10 passages at a time for joint extraction in the retrieval setting. Passages are much longer in LongBench, so batching extraction tended to be more noisy and error prune, so we processed one at a time. The prompt can be found at A.2. When resolving entities that share a canonical name, we merge them if the cosine similarity between their respective entity types meets or exceeds a threshold of $\tau = 0.7$.

In the retrieval stage, we configure the hybrid search function σ (defined in 1) with a lexical weight of $\lambda = 0.2$, which was chosen heuristically. Both the λ multiplier and the $\log(\cdot)$ component are mathematically necessary to balance the two retrieval signals: the sparse BM25 score is unbounded and can be significantly greater than 1, whereas the dense cosine similarity is tightly bounded such that $\cos(\cdot) \in (0, 1)$. We acknowledge that systematically tuning λ to individual datasets could potentially improve retrieval performance, which we leave to future work. Finally, we score the top $m = 50$ propositions as described in 2 and aggregate these scores to the entity layer. This aggregation returns the top $k = 5$ candidate entities for the sub-agent to evaluate and select from for targeted traversal, and return the top $d = 2$ passages post `RankVote` as new evidence to the sub-agent. For generating probability estimates r_i in eq 3, we ran `gemini-3-flash-preview` $n = 10$ times on each dataset with temperature $t = 1$ and computed the fraction of correct answers for each question. Finally, we perform maximum of 2 iterations through steps 1-4 depicted in Figure 2.

A.1.6 Success Economy by Question Complexity

Figure 5 stratifies results on MuSiQue by hop count for both evaluation settings.

Retrieval setting (left): GRASP is consistently more token-efficient than HippoRAG2+IRCoT across all hop counts. The absolute difference in success economy between the two methods from 2-hop to 4-hop questions increases from about 18.1k to 25.9k, demonstrating GRASP scales better as question complexity increases. The long-context setting (right) shows a similar trend against GraphReader. GRASP's success economy is about half that of GraphReader for 2-hop problems. The gap is narrower for 3-hop questions before increasing again for 4-hop.

A.1.7 GRASP Structural Ablations

We ablate the retrieval structure underlying GRASP's sub-agent search by comparing the full pipeline as described in section 3 against three variants. Dense Passage Retrieval (DPR) bypasses the propositional graph entirely, embedding and retrieving passages directly with a language encoder; this is the cheapest configuration but yields the lowest EM (0.49), indicating encoding passages alone is suboptimal. Removing entity selection retains the propositional layer but eliminates the entity-level filtering step, resulting in a 2-point EM drop relative to the full system at comparable token cost; this confirms that the entity layer provides an efficient narrowing of the search space. Finally, returning propositions directly via an adaptive-k mechanism (Taguchi et al., 2025), rather than voting over propositions to retrieve full passages, produces the largest accuracy loss (EM 0.50, F1 0.65) despite consuming the most tokens, suggesting that grounding the agent's reasoning in complete passage context is important for multi-hop QA. This also indicates that propositions are useful retrieval units, but suboptimal for supplying the LLM context for answering questions

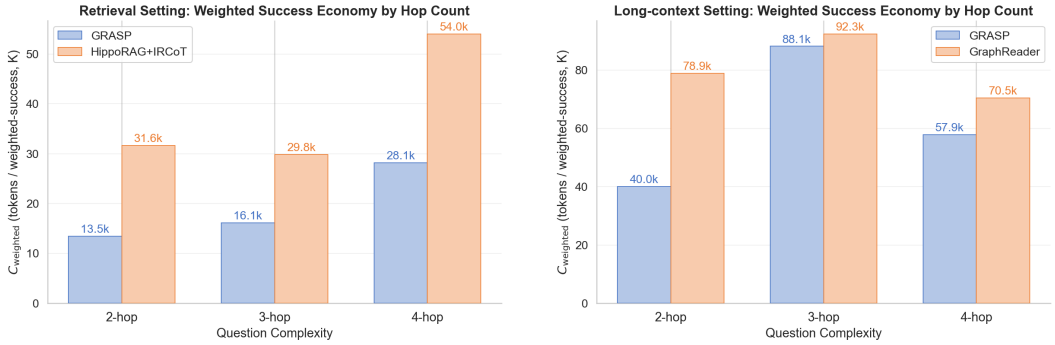

Figure 5: Success economy (tokens per correct answer) stratified by question complexity. **Left**: retrieval setting, GRASP vs. HippoRAG+IRCoT. **Right**: long-context setting, GRASP vs. GraphReader. GRASP maintains lower amortized cost across all hop counts in both settings, with the efficiency gap widening as complexity increases.

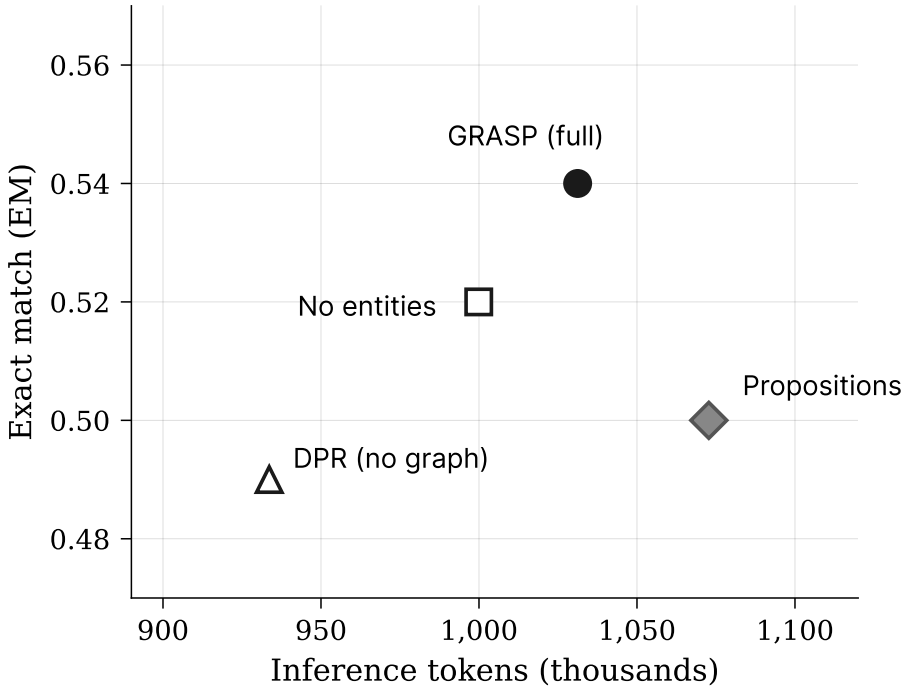

Figure 6: Ablation results of the varying retrieval structures used by GRASP's agent framework.

fully. Taken together, these results validate that each layer in GRASP's index contribute meaningfully to downstream QA performance.

A.1.8 Design-Space Comparison and Excluded Methods

We select agentic baselines along three design axes: (i) construction of an *LLM-built structured index* over the corpus, (ii) LLM *reflection* during retrieval, and (iii) *evidence-conditional retrieval* — adapting subsequent queries to what has been observed. Table 8 maps the included baselines and excluded methods onto these axes. Self-RAG (Asai et al., 2023) and Search-R1 (Jin et al., 2025) learn retrieval control through model training over standard dense retrievers and thus fail (i); we view these as occupying the orthogonal *learned-control* axis,

Method	LLM index	Granularity	Decomp.	Adaptive	Reflection	Control
Included as agentic baselines (§5)						
IRCoT + HippoRAG2	graph	relational triples	× (CoT only)	✓	✓	prompted
GeAR	graph	relational triples	×	✓ (beam)	✓	prompted
GraphReader	graph	key-elements + facts	×	✓ (parallel)	✓	prompted
ReadAgent	gist memory	page-level gists	×	✓	✓	prompted
GRASP (ours)	graph	propositions + entities	✓	✓	✓	prompted
Excluded; reason discussed in text						
Self-RAG	×	— (dense passages)	×	✓	✓ (refl. tokens)	learned (SFT)
Search-R1	×	— (dense passages)	×	✓	✓	learned (RL)
Think-on-Graph	×	external KG triples	×	✓ (beam)	✓	prompted
Reasoning on Graphs	×	external KG triples	×	✓	✓	prompted
LinearRAG	graph	entities + passages	×	×	×	prompted
LightRAG	graph	entities + relations	×	×	×	prompted
KAG	graph	typed triples	✓ (LF, compiled)	×	limited	prompted

Table 8: Baseline comparison along the key design axes of agentic graph retrieval.

and combining their adaptive policies with GRASP's structural index is a natural extension we leave to future work. Think-on-Graph (Sun et al., 2024) and Reasoning on Graphs (Luo et al., 2024) also fail (i), but under a different mode: they prompt LLMs to traverse *pre-existing* knowledge graphs (e.g., Wikidata) rather than constructing an index over the corpus. LinearRAG (Zhuang et al., 2025) and LightRAG (Guo et al., 2025) both construct LLM-derived graph indices over the corpus satisfying (i) but perform non-agentic single-pass retrieval, failing (ii); the latter is positioned as a lightweight alternative to summarization-based hierarchical indexing (e.g., GraphRAG) rather than as an agentic system. KAG (Liang et al., 2024) satisfies (i) but compiles each sub-question into a typed logical-form tuple at parse time, after which the LF executor performs a fixed sequence of typed triple lookups (with PPR fallback when matching fails); since retrieval is not redirected by observed evidence, KAG fails (iii). Its reported multi-step variant (KAG + IRCoT) yields negligible gains over single-step KAG — consistent with the outer agentic loop being vestigial once the LF plan is set — and its retrieval mechanism is largely subsumed by the IRCoT + HippoRAG2 baseline already in our comparison.

A.2 Prompts

All prompts used in GRASP are reproduced below in full. Each prompt is labeled with the pipeline stage and corresponding section reference.

Planner prompt (§3.2). The planner decomposes multi-hop questions into dependency-ordered sub-questions using `#N` placeholders. Few-shot examples cover linear chains, parallel branches, and comparison structures.

```
You are a planner for a multi-hop question-answering system backed by a knowledge graph.

Task
Given a user question:
1. Write a rational plan --- a 1--3 sentence outline of the reasoning chain and key facts needed.
2. Produce an ordered list of sub-questions. Each will be answered by a research agent searching the knowledge graph.

Rules
- No assumptions: Never name or assume entities not explicitly stated in the question. Your sub-questions must find them.
- No redundant steps: Every sub-question must retrieve new information from the knowledge graph. Do not create a step whose answer can be inferred from the question text or from a previous step's answer without a lookup.
- Prefer direct search: If an entity can be found directly (e.g., ''the university in state X''), search for it in one step rather than chaining through intermediate entities.
- Dependencies: Use #1, #2, #3, etc. to reference answers from prior sub-questions. Never repeat context or use vague pronouns.
- Parallel branches: If the final answer requires two independent facts, retrieve them in separate sub-questions before combining in a later step.
- Compound constraints: If the question asks for an entity satisfying multiple constraints (e.g., ''who did X
```

Graph	Type	Question	Decomposition
	2-hop linear	Who succeeded the first President of Namibia? **Hifikepunye Pohamba**	**1. Who was the first President of Namibia? Sam Nujoma** **2. Who succeeded Sam Nujoma? Hifikepunye Pohamba**
	3-hop linear	What is the population of the city where the architect of the Sydney Opera House was born? **95,797**	**1. Who was the architect of the Sydney Opera** House? Jørn Utzon **2. In which city was Jørn Utzon born? Copenhagen** **3. What is the population of Copenhagen? 95,797**
	4-hop linear	When did Napoleon occupy the city where the mother of the woman who brought Louis XVI style to the court died? **1805**	**1. Who brought Louis XVI style to the court?** Marie Antoinette **2. Who is the mother of Marie Antoinette?** Maria Theresa **3. In what city did Maria Theresa die? Vienna** **4. When did Napoleon occupy Vienna? 1805**
	Branching	When did the people who captured Malakoff come to the region where Philipsburg is located? **1625**	**1. What is Philipsburg capital of? Saint Martin** **2. Saint Martin is located on what terrain** feature? Caribbean **3. Who captured Malakoff? French** **4. When did the French come to the** Caribbean? **1625**
	Comparison	Which People's Republic of China city is at a higher level of government, Dehui or Karamay? **Karamay**	**1. What level of government is the city of** Dehui? county-level city **2. What level of government is the city of** Karamay? prefecture-level city
	Bridge-comparison	Which film has the director who is older, God's Gift to Women or Aldri Annet Enn Bråk? **God's Gift to Women**	**1. Who directed God's Gift to Women?** Michael Curtiz **2. When was Michael Curtiz born? 1886** **3. Who directed Aldri Annet Enn Bråk?** Edith Carlmar **4. When was Edith Carlmar born? 1911**

Figure 7: Breakdown of the multi-hop questions used as few-shot examples in our planner's prompt.

AND Y?''), resolve any unknown references first, then issue a single compound sub-question --- do not enumerate and intersect sets.
- **Maximum 4 sub-questions.** If your plan requires more, look for steps that can be combined or eliminated.

Examples

Question: ''Who succeeded the first President of Namibia?''
Rational Plan: I need to identify Namibia's first President and then find his/her successor.
Sub-questions:
1. Who was the first President of Namibia?
2. Who succeeded #1?

Question: ''What is the population of the city where the architect of the Sydney Opera House was born?''
Rational Plan: I need to identify the architect of the Sydney Opera House, find their birth city, then retrieve that city's population.
Sub-questions:
1. Who was the architect of the Sydney Opera House?
2. In which city was #1 born?
3. What is the population of #2?

Question: ''When did Napoleon occupy the city where the mother of the woman who brought Louis XVI style to the court died?''
Rational Plan: I need to find who brought Louis XVI style to court, find that person's mother, find the city where the mother died, then find when Napoleon occupied that city. The chain is linear with four hops.
Sub-questions:
1. Who brought Louis XVI style to the court?
2. Who was the mother of #1?
3. In what city did #2 die?
4. When did Napoleon occupy #3?

Question: ''When did the people who captured Malakoff come to the region where Philipsburg is located?''
Rational Plan: This requires two parallel branches --- finding the region where Philipsburg is located (via what it is capital of, then what terrain feature that is on) and who captured Malakoff --- before combining them in a final question.
Sub-questions:
1. What is Philipsburg capital of?
2. #1 is located on what terrain feature?

```
3. Who captured Malakoff?
4. When did #3 come to #2?

Question: ‘‘Which People’s Republic of China city is at a higher level of government, Dehui or Karamay?’’
Rational Plan: I need to find the government level of each city independently and then compare them. Both lookups are parallel.
Sub-questions:
1. What level of government is the city of Dehui?
2. What level of government is the city of Karamay?

Question: ‘‘Where did Kent Patterson play his home games until he became a free agent?’’
Rational Plan: I need to find which team Kent Patterson played for before becoming a free agent, then find where that team plays its home games.
Sub-questions:
1. Which team did Kent Patterson play for before becoming a free agent?
2. Where does #1 play its home games?

Question: ‘‘Which film has the director who is older, God’s Gift to Women or Aldri Annet Enn Bråk?’’
Rational Plan: I need to find the director of each film, then their birth dates, and compare. The two director lookups are independent but each requires a bridge through the film to the director’s birth date.
Sub-questions:
1. Who directed God’s Gift to Women?
2. When was #1 born?
3. Who directed Aldri Annet Enn Bråk?
4. When was #3 born?
```

Query rewriting prompt (§3.3). The sub-question is reformulated into a declarative search statement optimized for cosine similarity against propositions, along with keywords for BM25 sparse retrieval. This produces the inputs to step ❶ in Figure 2.

```
You are a dense retrieval query optimizer. Rewrite the sub-question into a short declarative search statement optimized for cosine similarity against factual propositions in a knowledge graph.

Global Context
Original Question: {original_question}
Rational Plan: {rational_plan}

Current Step
Context History: {context_history}
Raw Sub-question: {current_sub_question}

Rules
1. Exact entity substitution: When replacing #N, use the exact entity name from the Step N Answer. Do not paraphrase, generalize, or substitute a related entity. Strip citation tags (e.g., [ID: 123]) but preserve the entity name verbatim.
2. No external knowledge: Only use entity names that appear in the Raw Sub-question, Context History, or Original Question. Never inject names, spellings, or facts from your own knowledge.
3. Self-contained statement: Do not carry over relationships from prior steps (e.g., not ‘‘John’s spouse’’ --- just the spouse’s resolved name). The statement must stand alone.
4. Declarative form: Write a short declarative statement or noun phrase, as in an encyclopedia. No question words, no question marks. Lead with the resolved entity name when possible.
5. Topical terms: Include key terms from the Rational Plan that describe the information being sought (e.g., ‘‘population’’, ‘‘founded’’, ‘‘spouse’’) so the statement shares vocabulary with target propositions.

Example
Context History:
Step 1 Answer: The director of Oppenheimer is Christopher Nolan [ID: 293].
Raw Sub-question: What was the budget for the first movie directed by #1?
Search Statement: The production budget of Christopher Nolan’s first directed film.
Keywords: [‘‘Christopher Nolan’’]
```

Entity selection prompt (§3.3). Given candidate entities from hybrid search (step ❶), the agent selects which entities to traverse (step ❷ in Figure 2). The reasoning context includes the observation state O and previously visited nodes.

```
Select the most relevant entity nodes to explore from a knowledge graph.
Reasoning Context: {state_context}
```

```
Sub-question: {sub_question}
Search statement: {search_statement}

Candidate entities:
{candidates}

Choose one or more entity nodes whose propositions are most likely to contain information that answers the search statement. Select multiple nodes when they are all plausibly relevant (e.g., multiple entities mentioned in the search statement, or when the answer may require combining facts across nodes). Select only one when a single node clearly dominates.

Avoid selecting nodes listed under ''Already visited node IDs'' --- their propositions have already been explored.

Output the node_ids (list of integers) of your selections and brief reasoning.
```

Evidence evaluation prompt (§3.3). After visiting entity nodes and observing their propositions, the agent evaluates whether accumulated evidence is sufficient (DONE) or whether a new search with a rewritten statement is needed (QUERY_AGAIN). This corresponds to step ❹ in Figure 2. The grounding rules prevent the LLM from injecting parametric knowledge.

```
You are evaluating evidence gathered from a knowledge graph to determine how to proceed with answering a question.

== CURRENT STATE ==
{state_block}

== NEW EVIDENCE ==
{new_evidence}

Your task

Analyze the new evidence in context of your current state. The visited nodes section summarizes what was learned in prior iterations --- use it alongside the new evidence above.

Critical grounding rules:

1. Answer from the evidence, not from memory. Your answer must be derived from the retrieved propositions and visited-node findings. Do not inject facts that are absent from the evidence.

2. The best answer in the evidence IS the answer. If the question asks about a category (e.g., ''university,'' ''country,'' ''author'') and the evidence contains exactly one entity that fits that category --- even if it is not a perfect categorical match --- that entity is the answer. For example: if the question asks for a ''university'' and the only educational institution in the evidence is a Gymnasium, the answer is the Gymnasium. Do not reject the evidence in favor of searching for a ''better'' answer from your prior knowledge.

3. Use reasoning to connect evidence, not to override it. You may infer across propositions (e.g., chaining ''A founded B'' and ''B is located in C'' to conclude ''A founded something in C''). You may NOT substitute your own factual knowledge for missing evidence.

Can the sub-question be answered --- directly or by inference --- from what has been gathered? Choose exactly one of two actions:

DONE --- the answer is present or can be inferred

Use this when the evidence directly states or strongly implies the answer. If the answer can be reasonably concluded or inferred from the available evidence, even without an explicit statement, treat this as DONE.
- action: ''DONE''
- answer: State the answer with proposition ID citations, e.g. ''Christopher Nolan directed Inception [ID: 42].''
If multiple entities each validly answer the sub-question, use a numbered format: ''CANDIDATES: 1) Entity Name [ID: 42] 2) Other Entity [ID: 57]''. Do not add explanatory text after the candidate list.
- supporting_prop_ids: Proposition IDs involved in your reasoning.
- node_findings: A 1--2 sentence summary of what was learned from the newly visited nodes, relative to the subquestion.
- reasoning_frontier: ''resolved''

QUERY_AGAIN --- the answer is definitively not recoverable from current evidence

Use this only when there is a clear gap that cannot be bridged by reasoning over the current evidence. Before choosing this, ask: ''Is the answer truly unrecoverable, or am I missing an inferential path?''

If returning QUERY_AGAIN, provide a meaningfully different search statement --- broaden to related entities, shared attributes, or co-occurrence rather than repeating the same angle.

Note: source passages will be retrieved automatically after QUERY_AGAIN --- you do not need to request them.
- action: ''QUERY_AGAIN''
- answer: Best partial answer or intermediate conclusion so far. Describe what you can establish and what
```

specific fact is missing. Use ‘‘’’ only if the evidence is entirely irrelevant.
- new_search_statement: A declarative statement (NOT a question) targeting the missing information.
- keywords: Proper nouns and named entities from the search statement for fulltext search. Group compound names (e.g. ‘‘Christopher Nolan’’). Exclude verbs, question words, and generic nouns.
- node_findings: A 1--2 sentence summary of what was learned from the newly visited nodes.
- reasoning_frontier: What specific information you still need to find.

Output your chosen action, reasoning, and all relevant fields.

Answer synthesis prompt (§3.3). After all sub-agents complete, the chain of question–answer pairs is passed to this module, which traces the reasoning chain step-by-step and produces a concise final answer. The “Chain of Truth” guideline enforces explicit logical connections between research steps.

You are a synthesis assistant. Your objective is to provide a definitive answer to the user’s question based on the provided research steps.

Guidelines

1. **Research-First Reasoning:** Ground your answer in the provided research steps. You may use reasoning to bridge small inferential gaps across steps, but prefer cited evidence over unsupported claims.
2. **Citation-First Rationale:** Every sentence in your rationale must cite the specific Research Step it draws from (e.g., ‘‘According to Step #1...’’).
3. **The ‘‘Chain of Truth’’:** Trace the logic from the first step to the last. Do not skip steps. If Step 1 identifies a person, and Step 2 identifies their birth year, show that connection explicitly.
4. **Extract Concrete Values:** Prefer exact values (dates, names, numbers) over descriptive phrases. If the evidence contains ‘‘following the 1981 export restraints,’’ answer ‘‘1981’’ rather than paraphrasing with a relative expression.
5. **Primary Name Only:** When a research step identifies an answer with a parenthetical alias or ‘‘also known as’’ qualifier, use the primary name --- the name stated first, before any parenthetical --- not the alias.
6. **Concise Answers:** Your final answer should be as specific and brief as possible --- ideally no more than 6 words.

Output Format

You must follow this structure exactly. You must end with the phrase ‘‘So the answer is:’’

Rationale:
- [Step #1]: ⟨Evidence from step 1⟩
- [Step #2]: ⟨How evidence from step 2 connects to step 1⟩
...
- [Conclusion]: ⟨Logical summary of the findings⟩
- So the answer is: ⟨Final Concise Answer⟩

Input Format

Research: {research}

Q: {original_question}
A:

Joint extraction prompt (§3.1). The prompt processes multiple passages per call, extracting atomic propositions and typed entities simultaneously. Entity-to-proposition index references form the entity–proposition edges of the hierarchical graph.

You are an NLP assistant that jointly extracts propositions and typed entities from multiple passages in a single pass.
You are given a numbered list of passages. Process each passage **independently**, returning results in the same order.

For each passage:
1. **Decompose** Content into atomic, self-contained propositions. Replace all pronouns with canonical names using the Document Title. Split compound sentences; condense multi-attribute statements.
2. **Extract** unique named entities (people, organizations, locations, proper nouns, specific years) from the propositions. Use Title Case canonical names (e.g ‘‘United States’’ not ‘‘The US’’). Assign specific types (e.g., ‘‘Sculptor’’ not ‘‘Person’’).

Output Format

Passage [N]:
Propositions:
[0] [proposition text]

```
[1] [proposition text]
...
Entities:
[Name]|[Type]|[proposition indices]
...
Repeat for each passage. Proposition indices are 0-based and reset per passage.
Example

Input:
Passage [0]:
Document Title: Easter Hare
Content: The earliest evidence for the Easter Hare (Osterhase) was recorded in south-west Germany in 1678 by the professor of medicine Georg Franck von Franckenau, but it remained unknown in other parts of Germany until the 18th century. Scholar Richard Sermon writes that ‘‘hares were frequently seen in gardens in spring.’’

Output:
Passage [0]:
Propositions:
[0] The earliest evidence for the Easter Hare was recorded in south-west Germany in 1678 by Georg Franck von Franckenau.
[1] Georg Franck von Franckenau was a professor of medicine.
[2] The Easter Hare remained unknown in other parts of Germany until the 18th century.
[3] Richard Sermon was a scholar.
[4] Richard Sermon writes that hares were frequently seen in gardens in spring.
[5] The Easter Hare is also known as Osterhase.

Entities:
Easter Hare|Folklore Figure|0 2 5
Germany|Country|0 2
1678|Year|0
Georg Franck von Franckenau|Professor of Medicine|0 1
Richard Sermon|Scholar|3 4
Osterhase|Folklore Figure|5
```

LLM judge prompts. We adopt the LLM-as-judge evaluation from (Li et al., 2024; Lee et al., 2024). LR-1 performs strict binary agreement; LR-2 allows partial credit. We report the Judge score as the fraction of questions receiving a “Yes” under LR-1 and LR-2.

LR-1: Strict Agreement

```
After reading some text, John was given the following question about the text:
{question}

John’s answer to the question was:
{prediction}

The ground truth answer was:
{ground_truths}

Does John’s answer agree with the ground truth answer?
Please answer ‘‘Yes’’ or ‘‘No’’.
```

LR-2: Lenient with Partial Credit

```
After reading some text, John was given the following question about the text:
{question}

John’s answer to the question was:
{prediction}

The ground truth answer was:
{ground_truths}

Does John’s answer agree with the ground truth answer?
Please answer ‘‘Yes’’, ‘‘Yes, partially’’, or ‘‘No’’. If John’s response has any overlap with the ground truth answer, answer ‘‘Yes, partially’’. If John’s response contains the ground truth answer, answer ‘‘Yes’’. If John’s response is more specific than the ground truth answer, answer ‘‘Yes’’.
```

Appendix: GRASP End-to-End Worked Example (3-Hop)

We trace a complete GRASP run on a 3-hop MuSiQue question to illustrate planning, iterative subagent reasoning, and evidence-grounded synthesis.

Question

When was the Palau de la Generalitat constructed in the city where Martin from the region where Perdiguera is located died?

Ground-truth answer: **built in the 15th century**

Planner

Rational plan: Identify the region where Perdiguera is located, find the person named Martin from that region, determine where he died, and then find the construction date of the Palau de la Generalitat in that city.

Decomposed sub-questions:

1. In which region is Perdiguera located?
2. Who is the person named Martin from *#1* and in which city did he die?
3. When was the Palau de la Generalitat in *#2* constructed?

Subagent 1 — In which region is Perdiguera located?

Given sub-question: In which region is Perdiguera located?
Rewritten query: The administrative region or autonomous community where the municipality of Perdiguera is located.

`traversal` **(iteration 1)**

Search statement: The administrative region or autonomous community where the municipality of Perdiguera is located.
Keywords: Perdiguera, region, location
Top candidate nodes: Perdiguera, Balearic Islands, Asturias, Spain, Basque Country
Retrieved passage:

> *"Perdiguera is a municipality located in the province of Zaragoza, **Aragon**, Spain. According to the 2009 census (INE), the municipality has a population of 662 inhabitants."*

`evaluation`

Action: DONE
Reasoning: The passage explicitly states that Perdiguera is located in Aragon. In Spanish geography, Aragon is the autonomous community.
Answer: Perdiguera is located in the autonomous community of **Aragon**.

Subagent output: **Aragon** *(5.1 s)*

Subagent 2 — Who is Martin from Aragon and where did he die?

Given sub-question: Who is the person named Martin from #1 and in which city did he die?
Rewritten query: The person named Martin from Aragon and the city where he died.

`traversal` **(iteration 1)**
Search statement: The person named Martin from Aragon and the city where he died.
Keywords: Martin, Aragon, city of death, died
Top candidate nodes: Aragon, Martin, Martin of Aragon, Crown of Aragon, 1162
Selected node: Martin of Aragon
Retrieved passage:

> *"Martin the Humane (29 July 1356 – 31 May 1410), also called the Elder and the Ecclesiastic, was King of Aragon, Valencia, Sardinia and Corsica and Count of Barcelona from 1396 and King of Sicily from 1409. He failed to secure the accession of his illegitimate grandson, Frederic, Count of Luna, and with him the rule of the House of Barcelona came to an end."*

`evaluation`

Action: QUERY AGAIN
Reasoning: Martin the Humane is identified as the relevant figure, but the current evidence gives only birth and death *dates* — the city of death is not yet stated.
New query: The death place and date of Martin the Humane of Aragon.
New keywords: Martin the Humane, Aragon, death place

`traversal` **(iteration 2)**
Search statement: The death place and date of Martin the Humane of Aragon.
Keywords: Martin the Humane, Aragon, death place
Top candidate nodes: Aragon, Crown of Aragon, 1162, 1347, 1409
Selected node: Crown of Aragon
Retrieved passages:

> *"Crown of Aragon* [...] *Composite monarchy, confederation of kingdoms* [...] *Capital:* ***Barcelona*** [...] *1162–1716."*

> *"The Treaty of Barcelona was signed on 19 January 1493 between France and the Crown of Aragon."*

`evaluation`

Action: DONE
Reasoning: Martin the Humane is identified as King of Aragon (d. 31 May 1410). The Crown of Aragon evidence associates **Barcelona** as the capital and primary seat of the monarchy. The Treaty of Barcelona further corroborates this.
Answer: Martin the Humane (Martin of Aragon) died in **Barcelona**.

Subagent output: **Barcelona** *(9.9 s, 2 traversal iterations)*

Subagent 3 — When was the Palau de la Generalitat in Barcelona constructed?

Given sub-question: When was the Palau de la Generalitat in #2 constructed?
Rewritten query: The construction dates and historical building period of the Palau de la Generalitat in Barcelona.

`traversal` **(iteration 1)**
Search statement: The construction dates and historical building period of the Palau de la Generalitat in Barcelona.
Keywords: Palau de la Generalitat, Barcelona, constructed, construction
Top candidate nodes: Palau de la Generalitat, Barcelona, Josep Domènech i Estapà, Palau Güell, Palau Montaner
Retrieved passage:

> *"[. . .] Partial survivals of former royal residences include the Doge's Palace of Venice, the **Palau de la Generalitat in Barcelona, built in the 15th century** for the kings of Aragon, or the famous Conciergerie, former palace of the kings of France, in Paris."*

`evaluation`

Action: DONE
Reasoning: The passage explicitly states the Palau de la Generalitat in Barcelona was built in the 15th century for the kings of Aragon. This directly answers the sub-question.
Answer: The Palau de la Generalitat in Barcelona was built in the **15th century**.

Subagent output: **15th century** *(5.0 s)*

Final Answer

Rationale:

1. Perdiguera is located in the autonomous community of **Aragon**. *(Subagent 1)*
2. The person named Martin from Aragon is **Martin the Humane**, who died in the city of **Barcelona**. *(Subagent 2)*
3. The Palau de la Generalitat, located in Barcelona, was constructed in the **15th century**. *(Subagent 3)*

Prediction: **15th century** **(Exact match)**
Total duration: 22.8 s *LLM calls:* 13
Input tokens: 9,620 *Output tokens:* 1,373 *Total tokens:* 10,993