LOGICALLY PROVABLE REASONING FOR MULTI-HOP RETRIEVAL AUGMENTED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Complex questions cannot be reliably answered by a single query to a large language model (LLM) or even a large reasoning model (LRM). Solving such questions often requires multiple retrieval and reasoning steps, where some steps may involve evidence and intermediate results obtained earlier. The system must therefore handle competing reasoning paths, ambiguous entities, and potentially incorrect intermediate bindings. Existing retrieval-augmented generation (RAG) methods typically interleave retrieval and reasoning over multiple steps. However, document-level relevance does not reveal the logical role of each piece of evidence or guarantee that the resulting steps form a complete reasoning plan. To fill this gap, we propose **ProvableRAG**, a logic-based RAG framework that constructs a logically provable retrieval and reasoning plan to achieve conditional logical sufficiency for both retrieval and reasoning. Specifically, the framework first constructs a logical reasoning graph that specifies what information must be retrieved or inferred and how these steps are logically related to each other. A symbolic logic solver verifies whether a potential plan represented in the graph can derive the target answer, while the resulting proof structure and logical reasoning depth guide plan selection and execution. During execution, each step passes only the structured results required downstream, avoiding the accumulation of retrieved documents across nodes. By compiling formally verified proofs into executable retrieval-and-reasoning plans, ProvableRAG enables logically grounded and efficient multi-step evidence acquisition. Across three multi-hop QA benchmarks, ProvableRAG achieves the highest accuracy while significantly reducing search calls and token consumption compared with the strongest baselines. The code is available at `https://anonymous.4open.science/r/ICLR27_31209-BB20/`.

1 INTRODUCTION

Multi-hop question answering often requires acquiring facts whose relevance becomes clear only as intermediate entities and conclusions are resolved (Press et al., 2023a; Shi et al., 2026; Sun et al., 2026b). For example, answering the question *which of the two films, Titanic (1997) and Inception (2010), has the older director* requires linking each film to its director, obtaining the corresponding birth dates, and comparing them. Retrieving relevant passages about both films does not by itself establish these logical connections. The system must decide which facts to obtain and how to combine them while avoiding searches that do not advance the answer. These logically related decisions make multi-hop retrieval-augmented generation (RAG) a planning problem of retrieval and reasoning (Jeong et al., 2024; Verma et al., 2025; Wu et al., 2026).

Existing approaches aim to address two core problems in the planning decisions: *when to acquire additional evidence* and *how to organize relations between steps*. For evidence acquisition, iterative methods use intermediate reasoning to guide the next retrieval (Press et al., 2023a; Yao et al., 2023; Trivedi et al., 2023). Adaptive and learned controllers refine these decisions through retrieval gating, question-complexity routing, or reinforcement learning (Asai et al., 2023; Jeong et al., 2024; Jin et al., 2025; Wu et al., 2026). For relations, existing methods consider the dependency between steps, where explicit plans represent relations that would otherwise remain implicit in a sequential reasoning trace. A group of methods use dependency graphs to schedule subqueries and manage context (Verma et al., 2025; Chen et al., 2026a), while some others express workflows through

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

retrieval-inference strategies and executable programs, respectively (Chen et al., 2026b; Sun et al., 2026a). These approaches improve acquisition control and dependency management, but fail to reach a provability criterion: whether the information specified by a candidate plan jointly supports a derivation of the target answer.

A dependency-consistent execution order does not, on its own, establish a logically provable plan. A graph can be acyclic yet omit a premise required by the target or leave the logical relation between an intermediate result and the final answer unspecified. In the film example, retrieving two birth dates is insufficient unless each date is linked to the director of the corresponding film. Conversely, exploring additional biographical details can increase costs without contributing to the comparison. Neuro-symbolic approaches already use declarative specifications and logical solvers to establish conclusions (Pan et al., 2023; Olausson et al., 2023; Ye et al., 2023). Our key idea is to use a symbolic *proof of target derivability* to help the organization of retrieval and reasoning: the proof specifies how the planned evidence supports the target and which branches participate in that derivation. We apply this principle to evidence acquisition, where intermediate facts are not yet available when the plan is constructed. Realizing this idea requires verifying a plan before its evidence is available, then preserving the logical relations in the proof when translating it into executable operations.

We introduce **ProvableRAG**, a framework that uses symbolic proofs to guide multi-hop retrieval-and-reasoning planning. ProvableRAG constructs a logical reasoning graph of anticipated evidence, distinguishing operations that retrieve external facts from those that infer conclusions using upstream results. The graph nodes specify the inputs for retrieval or reasoning, output metadata, and the edges represent logical relations between steps. A symbolic logic solver Prover9 is adopted to check whether the anticipated evidence of a candidate plan would suffice to derive the target answer under the generated formal specification. A successful proof identifies a supporting derivation, and its structure guides plan selection. ProvableRAG compiles the selected proof into a retrieval-and-reasoning schedule. Independent nodes can execute in parallel, and dependent nodes consume the entity bindings or logical relations supplied by upstream nodes. In the film example, each director's birth date remains associated with the corresponding film before the comparison is performed. The framework prioritizes simpler verified plans with shallow logical proof depth and falls back to alternative verified plans when required evidence cannot be obtained. Restricting execution to the proved plans avoids exploring other candidate branches, while passing only the structured results needed downstream limits context growth across steps. We evaluate ProvableRAG on three multi-hop QA benchmarks and two web-search QA benchmarks. ProvableRAG achieves the highest accuracy among the evaluated baselines. Our contributions can be summarized as follows:

- We propose ProvableRAG, a logically provable planning framework that represents retrieval and reasoning dependencies explicitly and verifies whether anticipated evidence jointly supports the target answer under a formal question specification.
- We design a proof-guided execution mechanism that prioritizes plans by logical proof depth, preserves dependencies between retrieval and reasoning operations, and propagates only structured facts to control context growth.
- We conduct an empirical evaluation across three multi-hop QA benchmarks, and two web-search QA benchmarks, achieving the state-of-the-art highest accuracy.

2 Related Work

2.1 Multi-hop and Iterative RAG

Multi-hop RAG requires acquiring and combining evidence across interdependent reasoning steps. Methods address this problem through adaptive evidence acquisition and structured organization of retrieval and reasoning. For acquisition, Self-Ask, ReAct, and IRCoT interleave intermediate reasoning with follow-up queries or external actions (Press et al., 2023a; Yao et al., 2023; Trivedi et al., 2023). The resulting observations inform subsequent steps, allowing queries to depend on entities or facts discovered during execution. Self-RAG learns reflection tokens for retrieval and critique, while Adaptive-RAG selects retrieval strategies according to question complexity (Asai et al., 2023; Jeong et al., 2024). Search-o1 (Li et al., 2025) integrates search into extended reasoning and uses a separate Reason-in-Documents module to process retrieved material before incorporating it into the reasoning chain. Search-R1 and HiPRAG optimize search policies through outcome and process

supervision, respectively (Jin et al., 2025; Wu et al., 2026). Structured methods differ in whether they organize the evidence collection or the question-solving process. HopRAG constructs a passage graph with connections induced by pseudo-queries and explores it through a retrieve-reason-prune procedure (Liu et al., 2025a). For question-level planning, Plan*RAG organizes atomic subqueries into a DAG for parallel execution and bounded context use, while LogicRAG combines dependency-guided execution with graph and context pruning (Verma et al., 2025; Chen et al., 2026a). RT-RAG constructs reasoning trees through entity analysis and consensus-based selection, then gathers evidence through bottom-up traversal and query refinement (Shi et al., 2026). STRIDE separates retrieval from inference through an entity-agnostic strategy grounded during execution; PyRAG expresses retrieval workflows as programs with explicit variables and runtime repair (Chen et al., 2026b; Sun et al., 2026a). For this problem, ProvableRAG proposes proof-guided planning to check the joint sufficiency of anticipated evidence under a formal specification and restrict execution to operations supporting the selected derivation.

2.2 Neuro-Symbolic Reasoning with LLMs

Neuro-symbolic reasoning combines LLM-based semantic interpretation with explicit representations and external execution or deduction. Execution-based approaches, including PAL and Faithful Chain-of-Thought, generate intermediate representations whose operations are carried out by external executors (Gao et al., 2023; Lyu et al., 2023). This separates the generation of a reasoning procedure from the execution of its computational steps. Deduction-based approaches, including Logic-LM, LINC, and SatLM, translate problems into declarative specifications for automated solvers (Pan et al., 2023; Olausson et al., 2023; Ye et al., 2023). In particular, LINC delegates first-order deduction to Prover9, whereas SatLM expresses constraints whose solutions are determined by a satisfiability solver. Symbolic information can also augment neural reasoning: Logic-of-Thought derives additional propositional consequences and translates them back into text to supplement the original context (Liu et al., 2025b). For planning, LLM+P translates natural-language problem descriptions into PDDL and delegates plan search to a classical planner (Liu et al., 2023), while LLM-Modulo uses external verifiers to assess candidates and guide their refinement (Kambhampati et al., 2024). Other approaches combine formal structure with textual knowledge: KAG combines logical-form-guided reasoning with knowledge graphs and text retrieval (Liang et al., 2025), while Logitext represents partially structured documents as natural-language text constraints and interleaves LLM-based constraint evaluation with satisfiability modulo theory solving (Oh et al., 2026). Across these approaches, the reliability of symbolic conclusions depends on the fidelity of semantic translation and the choice of intermediate representation (Olausson et al., 2023; Beiser et al., 2025). ProvableRAG's formulation uses conditional proofs to turn target derivations into acquisition plans, making proof dependencies available to guide retrieval before the required evidence is obtained.

3 Problem Formulation

Given a question q and a document collection $\mathcal{D}$, we seek an evidence-supported answer a. Formalizing q yields an answer condition $Q_q(a)$, meaning "a answers q," and a fixed background theory Γ_q containing the given facts and admitted inference rules. Facts to be retrieved are not in Γ_q.

Solving q composes retrieval and reasoning operations. Let F_t contain the ground facts accepted after t acquisition operations, where a ground fact has no unresolved variables. Initially $F_0 = \varnothing$, since given facts remain in Γ_q. Each operation performs retrieval or reasoning on evidence, applies an admitted rule, or performs a checked computation, then updates

$$F_{t+1} = F_t \cup \{h \mid h \text{ is accepted from the (t+1)-th operation}\}. \tag{1}$$

Operations may combine several earlier facts, and derived results may supply inputs to subsequent retrieval. After T acquisition operations, including unsuccessful attempts, successful completion requires a concrete answer a satisfying

$$\Gamma_q \cup F_T \vdash Q_q(a). \tag{2}$$

Here $\vdash$ denotes logical derivability under the formal specification. This criteria requires that the retrieved or derived facts combined with Γ_q can lead to a complete proof of the answer.

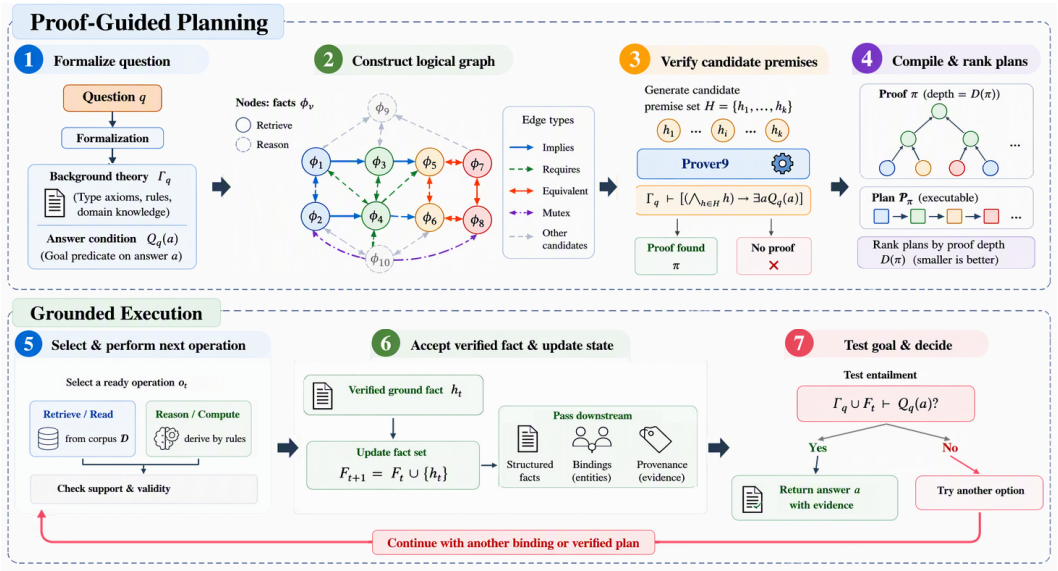


Figure 1: The framework of ProvableRAG. AND/OR groups require all/any prerequisites; execution checks support, grounding, scope, and consistency. Success returns the answer with proof and provenance; failure triggers alternative bindings, operations, or verified plans.

4 METHOD

The central idea of ProvableRAG is to use a proof to decide what evidence to acquire. Before retrieval, the system checks whether the facts specified by a candidate plan would jointly establish the answer condition in Equation (2). During execution, it obtains and checks those facts, grounding the plan's variables with entity or value bindings supported by evidence. This separates two questions that document relevance alone cannot resolve: *would this information be sufficient?* and *has this information actually been established?* We first construct a graph of required facts and their relations, use conditional proofs to select an executable plan, and then ground the proof through retrieval and reasoning. Appendix C provides a worked example of conditional proof checking, depth-guided plan selection, and grounded execution.

4.1 REPRESENTING RETRIEVAL AND REASONING IN LOGICAL GRAPHS

Representing the missing information. The graph is organized around facts because the same fact may be obtained from a passage or inferred from earlier results. Given q, an LLM first identifies the required information and then proposes relations among the resulting nodes. Keeping these two stages separate allows relations to refer to a fixed set of information needs (Appendix F.1). The resulting graph is $G_q = (V, E)$, where each node v represents a fact to establish, written as an atomic formula ϕ_v. Its record specifies the allowed acquisition modes, RETRIEVE and/or REASON and the required inputs. Unknown values remain variables until evidence supplies them.

For the film comparison, $\mathrm{Director}(f_i, d_i)$ asks for the director of film f_i, and $\mathrm{Born}(d_i, b_i)$ asks for that director's birth date, for $i \in \{1, 2\}$. The shared variable d_i carries the identity from one step to the next. A node therefore records more than an isolated fact: it preserves which earlier results make the retrieval or reasoning possible.

Logical relations. The edges in E express four types of relation:

- REQUIRES(u, v): the result of u is needed before executing v, as when a director identity supplies a birth-date query. This dependency does not imply that ϕ_u proves ϕ_v.
- IMPLIES(u, v): a rule allows $\phi_u \rightarrow \phi_v$, so establishing u can avoid acquiring v separately.

- EQUIVALENT(u, v): the two facts support inference in either direction, providing alternative ways to establish the needed information.
- MUTEX(u, v): the two nodes cannot be accepted together on one route. Different sources or complementary facts, such as the two directors' birth dates, do not constitute a conflict.

All four types guide plan construction; only admitted inference rules support deduction in Γ_q, and proposing an edge does not add a new rule to the theory. In particular, REQUIRES and MUTEX govern admissible execution without adding an implication or a negated fact.

Combining facts. All REQUIRES predecessors are mandatory, so two such edges make the comparison wait for both birth dates. Additional prerequisite groups specify alternative input combinations through CONJUNCTION within each group and DISJUNCTION between groups. Rules that combine facts use a conjunctive antecedent rather than separate unary IMPLIES edges: neither comparison input alone implies the result. The proposed goal groups specify combinations of nodes intended to answer all parts of q, while the next stage checks whether those combinations actually support a derivation.

4.2 VERIFYING PLANS BEFORE RETRIEVAL

Checking sufficiency without assuming truth. Starting from a proposed goal group, we select a producer for each required fact and recursively include its inputs. Alternative acquisition modes and prerequisite groups give different candidate plans; incompatible node combinations are discarded. Let $\mathcal{H}$ contain the candidate's anticipated retrieval facts and checked-computation outcomes, whose values may still be unknown. Each premise must have a specified producer and a feasible chain of input suppliers. Rule-derived intermediate conclusions and the answer itself must follow by deduction, rather than being placed in $\mathcal{H}$ as assumptions. Using the same fixed Γ_q as in Equation (2), Prover9 (Olausson et al., 2023) searches for a proof π of

$$\Gamma_q \vdash \left[\left(\bigwedge_{h \in \mathcal{H}} h \right) \longrightarrow \exists a \, Q_q(a) \right], \tag{3}$$

The bracketed implication is interpreted under universal closure, with all free variables implicitly universally quantified. The guarantee is conditional: for every assignment to the premise variables, if all premises hold, an answer exists. Unlike the accepted facts in F_t, the premises in $\mathcal{H}$ need not yet be ground or supported by evidence, and planning does not add them to F_t.

For example, one branch of the film comparison uses the rule

$$\mathrm{Director}(f_1, d_1) \wedge \mathrm{Born}(d_1, b_1) \wedge \mathrm{Director}(f_2, d_2) \wedge \mathrm{Born}(d_2, b_2) \wedge \mathrm{Earlier}(b_1, b_2) \;\rightarrow\; Q_q(f_1).$$

Planning establishes that these linked facts would suffice to answer f_1. Retrieval must still supply the identities and dates, and a comparison must establish $\mathrm{Earlier}(b_1, b_2)$. The opposite comparison supports a branch for f_2. Removing either director link breaks this derivation even if two dates are available. This is the role of the proof check: it tests the connections needed for the answer, beyond whether the proposed searches can run in a valid order.

Turning a proof into a plan. To compile a proof, we require it to identify the rule instances and an answer expression that can be grounded during execution. We trace the proof backward from that answer and retain the operations that produce its used premises and implement its rule applications. We then add the required input suppliers, mandatory predecessors, and one admissible group for each alternative prerequisite. The resulting plan P_π must have acyclic execution dependencies and satisfy the graph's compatibility constraints. For an EQUIVALENT relation, only the direction used in the derivation becomes an execution dependency. Thus, the candidate graph may contain cycles while the selected plan is executable. The plan omits operations that contribute neither to the proof nor to its required inputs.

Verified plans are tried in increasing proof depth $D(\pi)$, the largest number of logical rule applications from a given fact or open premise to the answer; ties follow candidate order. This preference favors shorter deductive chains, while allowing a proof premise to require several retrieval steps. It is not a minimum-retrieval-cost guarantee. A candidate for which no proof is found within the planning budget remains unverified.

4.3 GROUNDING THE PLAN IN EVIDENCE

Acquiring and checking a fact. Execution progressively replaces anticipated premises with established facts. A node is ready when its selected prerequisites are resolved and their results provide consistent bindings. In RETRIEVE mode, the executor uses these inputs to instantiate a query, retrieves passages from $\mathcal{D}$, and extracts a candidate fact. In REASON mode, it applies an admitted rule or the recorded computation to established premises. The query and candidate-generation prompts are given in Appendix F.2.

A candidate is accepted only after checking its support, entity identity, scope, and compatibility with established facts under Γ_q. The accepted ground instance of ϕ_v updates the fact state through Equation (1), while evidence identifiers and supporting premise nodes are retained as provenance. Ambiguous or conflicting candidates remain outside the fact state. In particular, a planned comparison becomes a fact only when its outcome is checked against the acquired inputs.

Passing the information needed downstream. The proof identifies both which facts are needed and where they will be used. Each dependent operation receives those facts and bindings together with provenance identifiers. Raw passages remain available for evidence checking, while only structured results needed by the next operation are transferred. All uses of a shared proof variable must agree on its binding; alternative assignments are evaluated separately rather than mixed within a derivation. Independent ready nodes need no artificial ordering. Algorithm 1 presents a serial schedule.

Completing or revising the execution. Before the first operation and after each fact-state update, Prover9 checks whether $\Gamma_q \cup F_t \vdash Q_q(a)$ for a concrete a. Grounding the premises used by a compiled proof under one consistent assignment supplies the derivation required by Equation (2). The executor then returns a with its provenance. Otherwise, it selects a ready operation from the first viable plan in increasing $D(\pi)$. Failed attempts or outcomes inconsistent with an anticipated premise trigger another compatible binding or a fallback to the next verified plan. Attempt records prevent repeating an exhausted option, and accepted facts remain reusable across plans when their scope and bindings match. If no viable operation fits the remaining budget, the result is unresolved. The guarantee is relative to the formal specification and accepted evidence: the symbolic proof checks the derivation, while factual correctness depends on grounding its premises faithfully.

4.4 WHY PROOF GUIDANCE CAN REDUCE COST

Proof guidance reduces unnecessary work in two places. First, execution is restricted to the operations supporting a selected derivation and their input dependencies. This can avoid searches for related facts that do not contribute to the answer, without asserting that the selected plan is globally minimal. Second, downstream operations consume the structured results identified by those dependencies, reducing repeated processing of the same passages.

For a chain of n model calls whose first $n-1$ calls each read L new evidence tokens, forwarding all earlier passages transfers $n(n-1)L/2$ tokens. Passing at most ℓ tokens of structured results per step reduces this term to at most $(n-1)\ell$; initial reading costs remain. Here n counts model calls, which may occur several times within one acquisition operation. These savings require bounded messages and must exceed planning overhead to reduce total input tokens. Appendix E derives the general dependency-graph comparison and states its assumptions.

5 EXPERIMENTAL RESULTS

We evaluate ProvableRAG in terms of answer accuracy and evidence acquisition cost. Our experiments compare its performance with existing RAG methods, examine the contributions of its components, and analyze token consumption, search calls, and response time. We further study how retrieval settings affect answer accuracy and computational cost.

5.1 EXPERIMENTAL SETUP

Datasets. We evaluate ProvableRAG on three multi-hop question-answering benchmarks: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), abbreviated as 2WikiMQA, and the answerable subset of MuSiQue (Trivedi et al., 2022). Following the corpus-construction pro-

Table 1: Results on three multi-hop QA benchmarks. Str. and LLM denote string-based and LLM-based accuracy (%). Bold and underlined values denote the best and second best results, respectively.

Method	Closed-source backbone GPT-4O-MINI						Open-source backbone MISTRAL-SMALL-4					
	HotpotQA		2WikiMQA		MuSiQue		HotpotQA		2WikiMQA		MuSiQue	
	Str.	LLM	Str.	LLM	Str.	LLM	Str.	LLM	Str.	LLM	Str.	LLM
Base model	38.7	36.3	26.4	24.3	17.6	14.0	34.4	39.1	22.1	20.8	9.7	10.6
Vanilla RAG (Top-3)	43.2	53.1	43.0	42.0	20.3	23.6	39.3	45.2	37.8	39.6	15.3	18.9
G-Retriever	28.5	40.9	26.7	35.7	9.1	15.6	27.4	34.5	22.5	30.9	7.8	13.4
KGP	46.4	57.1	47.5	43.7	23.3	27.5	41.5	52.9	42.4	42.9	15.4	22.1
GraphRAG	39.6	45.2	46.3	43.3	16.5	23.1	33.8	40.5	41.6	37.8	10.8	18.5
RAPTOR	48.1	57.8	47.7	45.9	25.2	29.1	43.4	51.2	43.7	43.9	17.6	24.5
HippoRAG2	56.7	61.9	50.0	47.1	27.0	32.6	47.9	53.8	44.9	47.3	19.9	25.8
LogicRAG	54.8	62.6	64.7	62.5	30.4	37.5	50.5	56.3	56.6	60.1	24.9	31.2
ProvableRAG (ours)	**59.2**	**65.8**	**66.3**	**68.6**	**32.1**	**37.6**	**53.7**	**57.8**	**60.9**	**63.8**	**31.7**	**34.7**

cedure used by IRCoT (Trivedi et al., 2023) and HippoRAG (Gutiérrez et al., 2025) , we gather all candidate passages associated with the selected questions, including supporting and distractor passages, to construct a retrieval corpus for each dataset. We additionally evaluate ProvableRAG on BrowseComp+ (Chen et al., 2025) and Bamboogle (Press et al., 2023b), with answer quality and search costs reported in Appendix Table 5.

Evaluation metrics. For the three multi-hop QA benchmarks, we report string-based accuracy (Str.) and LLM-judged accuracy (LLM). String-based accuracy checks whether the normalized response contains the reference answer, while LLM-judged accuracy evaluates answer correctness using a language model judge. Efficiency is measured by mean token consumption, search calls, and response time per question. Token consumption is expressed in tokens.

Baselines. We compare against the following baselines: (i) direct LLM generation; (ii) Vanilla RAG (Lewis et al., 2020) with top-3 retrieval; (iii) structured retrieval methods, including G-Retriever (He et al., 2024), KGP (Wang et al., 2024), GraphRAG (Edge et al., 2025), RAPTOR (Sarthi et al., 2024), HippoRAG2 (Gutiérrez et al., 2025), and LogicRAG (Chen et al., 2026a). For BrowseComp+ and Bamboogle, we use direct generation, Vanilla RAG (Lewis et al., 2020), ReAct (Yao et al., 2023), and LogicRAG (Chen et al., 2026a).

5.2 MAIN RESULTS

Table 1 compares ProvableRAG with the baselines on HotpotQA, 2WikiMQA, and MuSiQue using string-based and LLM-judged accuracy. ProvableRAG achieves the highest scores under both metrics on all three benchmarks with GPT-4O-MINI and MISTRAL-SMALL-4.

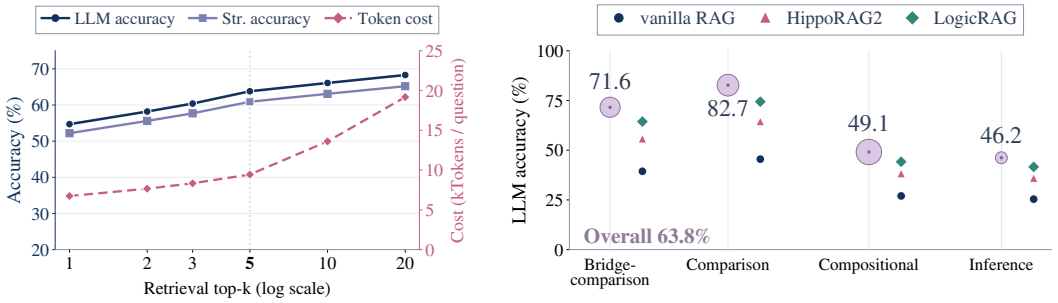
Zero-shot LLM inference. Direct generation achieves limited accuracy across the three benchmarks, with both backbones performing worst on MuSiQue. The consistent gap to Vanilla RAG highlights the benefit of grounding answers in external evidence.

Graph-based RAG. HippoRAG2 achieves the strongest results among methods based on preconstructed corpus graphs. LogicRAG, which constructs question-specific dependency graphs, surpasses these methods in LLM-judged accuracy across all three datasets and both backbones.

ProvableRAG. ProvableRAG improves on LogicRAG under both metrics in every setting in Table 1, with the largest gains in LLM-judged accuracy on 2WikiMQA for both backbones. We further extend the multi-hop QA evaluation to GPT-5.6-LUNA-MEDIUM and GEMMA-4-31B-IT, with results reported in Appendix Table 6.

5.3 EFFICIENCY AND MULTI-HOP PERFORMANCE

Efficiency analysis. Figure 2(a) examines the effect of retrieval top-k with MISTRAL-SMALL-4 on 2WikiMQA, showing answer accuracy on the left axis and token consumption on the right. Increasing k from 1 to 5 improves both string-based and LLM-judged accuracy. Further increases continue to improve performance, but the additional gains become smaller relative to the growing token cost. In particular, increasing k from 5 to 20 approximately doubles token consumption. The



(a) Top-k

(b) Question type

Figure 2: Analysis on 2WikiMQA with MISTRAL-SMALL-4. (a) String-based accuracy, LLM-judged accuracy, and token consumption across top-k settings. (b) LLM-judged accuracy across question types. Purple circles denote ProvableRAG with the size indicating number of questions.

declining accuracy gain per additional token supports our default setting of $k = 5$ as a practical balance between answer quality and processing cost. Corresponding analyses on HotpotQA and MuSiQue are provided in the appendix. The comparison with LogicRAG provides complementary evidence: ProvableRAG achieves higher accuracy under both metrics across all three benchmarks while consuming fewer online tokens with GPT-4O-MINI and MISTRAL-SMALL-4 (Tables 1 and 2). These results are consistent with our design of acquiring facts required by a verified plan and passing compact intermediate results between reasoning steps. This design limits repeated processing of retrieved passages while preserving the information needed for subsequent reasoning.

Multi-hop reasoning capability. Figure 2(b) shows the strongest performance on comparison questions, followed by bridge-comparison questions. The latter require resolving intermediate entities before comparing the associated facts, making them informative about the system's ability to connect dependent retrieval and reasoning steps. Strong performance on these questions is consistent with our explicit representation of prerequisite facts and their dependencies. Accuracy is lower on compositional and inference questions, identifying these reasoning patterns as remaining challenges. Corresponding analyses on HotpotQA and MuSiQue with MISTRAL-SMALL-4 are provided in Figure 3 in the appendix.

Table 2: Averaged time and token consumption on 2WikiMQA with GPT-4O-MINI.

Method	Time (s) ↓	Tokens ↓
GPT-4o-mini	5.88	216.2
Vanilla RAG (Top-3)	4.28	489.7
G-Retriever	12.50	1000.0
KGP	70.72	11097.8
GraphRAG	13.05	4699.8
RAPTOR	5.79	2568.0
HippoRAG2	5.89	2809.2
LogicRAG	9.83	1777.9
ProvableRAG (ours)	11.50	1531.2

5.4 ABLATION STUDY

Reasoning depth improves the accuracy–efficiency trade-off. Table 3 compares four scheduling strategies with MISTRAL-SMALL-4 while keeping all other components fixed. Reasoning-depth-guided scheduling achieves the highest LLM-judged accuracy across all three benchmarks. Coverage-plus-cost is the closest alternative in accuracy, but consumes more tokens on every dataset. Random selection uses the fewest tokens, accompanied by lower accuracy, while path-length selection produces the lowest accuracy and the highest token consumption. The contrast between reasoning depth and path length suggests that favoring shorter completed dependency chains does not necessarily improve reasoning. Using reasoning depth as the scheduling criterion provides a more favorable trade-off, improving accuracy over all alternatives while reducing token consumption relative to coverage-plus-cost and path-length selection.

Logical graph construction is central to answer quality. Table 4 evaluates all eight ablation combinations of logical graph construction (A), reasoning-depth-guided scheduling (B), and structured inter-step information transfer (C). ✓ indicates an enabled component and × a disabled component. All differences are relative to the full model (Comb. h). The full model achieves the highest accu-

Table 3: Planning ablation with MISTRAL-SMALL-4.

Strategy	**HotpotQA**		**2WikiMQA**		**MuSiQue**	
	LLM ↑	kTokens ↓	LLM ↑	kTokens ↓	LLM ↑	kTokens ↓
Coverage + cost	<u>57.0</u>	9.8	<u>62.9</u>	9.9	<u>31.0</u>	11.2
Random	51.3	**8.7**	52.6	**7.4**	22.5	**8.1**
Path length	49.5	12.4	50.4	11.7	20.3	18.6
Reasoning depth (ours)	**57.8**	<u>9.52</u>	**63.8**	<u>9.44</u>	**34.7**	<u>9.49</u>

Table 4: Component ablation on three multi-hop QA benchmarks with MISTRAL-SMALL-4.

				HotpotQA		**2WikiMQA**		**MuSiQue**	
Comb.	**A**	**B**	**C**	ΔAcc.	ΔkTok.	ΔAcc.	ΔkTok.	ΔAcc.	ΔkTok.
(a)	×	×	×	−43.7	+3.59	−39.3	+4.41	−27.9	+12.11
(b)	×	×	✓	−45.2	+0.54	−44.9	+0.62	−27.3	+4.51
(c)	×	✓	×	−15.9	+3.48	−16.1	+3.59	−23.8	+9.82
(d)	×	✓	✓	−2.4	+0.08	−2.5	+0.26	−7.6	+2.61
(e)	✓	×	×	−4.3	+3.55	−7.7	+4.03	−5.4	+8.76
(f)	✓	×	✓	−0.8	+0.28	−0.9	+0.46	−3.7	+1.71
(g)	✓	✓	×	−1.5	+3.28	−1.6	+3.46	−5.7	+7.31
(h)	✓	✓	✓	**0.0**	**0.00**	**0.0**	**0.00**	**0.0**	**0.00**

racy and lowest token consumption across all configurations on every benchmark. Among single-component removals, disabling the logical graph (Comb. d) causes the largest accuracy loss. The graph-only configuration (Comb. e) also achieves higher accuracy than scheduling alone (Comb. c) or structured transfer alone (Comb. b), supporting the central role of logical graph construction in the framework. We also conduct the full component ablation with GPT-4O-MINI, observing the same accuracy and token-cost trends under single-component removal (See Appendix Table 7).

Scheduling and structured transfer complement the logical graph. Removing depth-guided scheduling (Comb. f) reduces accuracy and increases token consumption across all three datasets. Removing structured transfer (Comb. g) likewise degrades both metrics and causes the largest token increase among single-component removals. These effects, together with those of removing the logical graph, are most pronounced on MuSiQue. The configurations without either graph construction or scheduling (combs. a and b) yield the lowest accuracy. Adding structured transfer alone (Comb. a to Comb. b) reduces token consumption but has mixed effects on accuracy. When combined with scheduling (Comb. c to Comb. d), graph construction (Comb. e to Comb. f), or both (Comb. g to Comb. h), structured transfer consistently improves accuracy and lowers token consumption. Together, these results support our core design: logical graph construction supports answer quality, while depth-guided scheduling further improves the effectiveness and efficiency of execution.

6 CONCLUSION

In this paper we present **ProvableRAG**, a framework that formulates multi-hop retrieval-augmented generation as the construction and execution of a logically provable plan. Rather than relying only on document relevance or dependency-consistent execution, ProvableRAG represents anticipated evidence and its logical relations in a reasoning graph, verifies whether candidate premises are jointly sufficient to derive the target answer, and compiles the resulting proof into an executable retrieval-and-reasoning schedule. Experiments across multi-hop and web-search question-answering benchmarks show the superiority of the framework. The results demonstrate that symbolic proofs can serve not only as post hoc validators of reasoning but also as practical structures for organizing the reasoning of LLMs. ProvableRAG nevertheless has several limitations. Like other neuro-symbolic systems, symbolic verification establishes the validity of a derivation, but the semantic validation and evidence correctness still depend on the LLMs and databases. Moreover, we extend the framework from RAG to web-search benchmarks, but the current evaluation still focuses on question answering. Future work can introduce more evaluation on more general agentic tasks, with dynamic environments requiring more kinds of tools or skills.

AI USE STATEMENT

In this work, we used generative AI tools to assist with editing the research paper to improve readability. We also used generative AI tools as backbones in our empirical analysis. We have not used generative AI tools for writing proofs, implementing methods, interpreting results, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, supporting qualitative and thematic data analysis. Generating synthetic datasets, assisting with translation and cleaning and reformatting datasets are not applicable to this work. Additionally, we used generative AI tools for searching for information and formatting references. We have reviewed all AI-assisted work. We reviewed the reference formatting generated or revised by AI. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of our proposed approach, experimental settings, and implementation details in the main paper and appendix. An anonymous version of the source code and the full implementation are provided in the supplementary materials.

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.

Alexander Beiser, David Penz, and Nysret Musliu. Intermediate languages matter: Formal choice drives neurosymbolic llm reasoning. 2025. URL https://arxiv.org/abs/2502.17216.

Shengyuan Chen, Chuang Zhou, Zheng Yuan, Qinggang Zhang, Zeyang Cui, Hao Chen, Yilin Xiao, Jiannong Cao, and Xiao Huang. You don't need pre-built graphs for RAG: Retrieval augmented generation with adaptive reasoning structures. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(36):30270–30278, March 2026a. ISSN 2374-3468. doi: 10.1609/aaai.v40i36.40278. URL https://ojs.aaai.org/index.php/AAAI/article/view/40278.

Wei Chen, Lili Zhao, Zhi Zheng, HuiJun Hou, and Tong Xu. Stride: Strategic iterative decision-making for retrieval-augmented multi-hop question answering. 2026b. URL https://arxiv.org/abs/2604.17405.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. 2025. URL https://arxiv.org/abs/2508.06600.

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/2404.16130v2.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

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 Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21497–21515. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/gutierrez25a.html.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models, 2025. URL https://arxiv.org/abs/2405.14831.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 132876–132907. Curran Associates, Inc., 2024. doi: 10.52202/079017-4224. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/efaf1c9726648c8ba363a5c927440529-Paper-Conference.pdf.

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.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)*, pp. 7036–7050, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 22895–22907. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/kambhampati24a.html.

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. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025.

Lei Liang, Zhongpu Bo, Zhengke Gui, Zhongshu Zhu, Ling Zhong, Peilong Zhao, Mengshu Sun, Zhiqiang Zhang, Jun Zhou, Wenguang Chen, et al. Kag: Boosting llms in professional domains via knowledge augmented generation. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 334–343, 2025.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. Hoprag: Multi-hop reasoning for logic-aware retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 1897–1913, 2025a.

Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Yuting Zeng, Jiaxing Wang, Xingyu Wang, Hailong Yang, and Jing Li. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10168–10185, 2025b.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In *Proceedings of the 13th international joint conference on natural language processing and the 3rd conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long papers)*, pp. 305–329, 2023.

Hyunseok Oh, Sam Stern, Youngki Lee, and Matthai Philipose. Neurosymbolic language reasoning as satisfiability modulo theory. *arXiv preprint arXiv:2602.18095*, 2026.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, 2023.

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

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

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

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 32628–32649, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8a2acd174940dbca361a6398a4f9df91-Paper-Conference.pdf.

Yuling Shi, Maolin Sun, Zijun Liu, Mo Yang, Yixiong Fang, Tianran Sun, and Xiaodong Gu. Reasoning in trees: Improving retrieval-augmented generation for multi-hop question answering. In *Companion Proceedings of the ACM Web Conference 2026*, pp. 626–635, 2026.

Jiashuo Sun, Jimeng Shi, Yixuan Xie, Saizhuo Wang, Jash Rajesh Parekh, Pengcheng Jiang, Zhiyi Shi, Jiajun Fan, Qinglong Zheng, Peiran Li, et al. Retrieval is cheap, show me the code: Executable multi-hop reasoning for retrieval-augmented generation. *arXiv preprint arXiv:2605.12975*, 2026a.

Jingbo Sun, Wenyue Chong, Songjun Tu, Qichao Zhang, Yaocheng Zhang, Jiajun Chai, Xiaohan Wang, Wei Lin, Guojun Yin, and Dongbin Zhao. Autosearch: Adaptive search depth for efficient agentic rag via reinforcement learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 28059–28079, 2026b.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multi-hop 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. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. Plan$ˆ\ast$RAG: Efficient test-time planning for retrieval augmented generation. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=gi9aqlYdBk.

Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. Knowledge graph prompting for multi-document question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19206–19214, March 2024. ISSN 2374-3468. doi: 10.1609/aaai.v38i17.29889. URL https://ojs.aaai.org/index.php/AAAI/article/view/29889.

Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. Hiprag: hierarchical process rewards for efficient agentic retrieval augmented generation. In *International Conference on Learning Representations*, volume 2026, pp. 82317–82347, 2026.

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. *CoRR*, abs/1809.09600, 2018. URL http://arxiv.org/abs/1809.09600.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. *Advances in Neural Information Processing Systems*, 36:45548–45580, 2023.

A Experimental Settings and Evaluation

A.1 Benchmarks and Retrieval Corpora

The multi-hop QA evaluation uses HotpotQA, 2WikiMQA, and the answerable subset of MuSiQue. For each dataset, the retrieval corpus contains all candidate passages associated with the selected questions, including supporting and distractor passages, following the procedure described in the main experimental setup. BrowseComp+ (Chen et al., 2025) and Bamboogle (Press et al., 2023b) are web-search QA benchmarks that provide additional searching protocols. Results on these two benchmarks are presented in Appendix B.1.

A.2 Backbones and Comparison Settings

Table 1 reports multi-hop QA results with GPT-4o-mini and Mistral-Small-4. Table 6 adds GPT-5.6-luna-medium and Gemma-4-31B-IT, and includes the Mistral-Small-4 results for reference. The BrowseComp+ and Bamboogle comparison uses GPT-5.6-luna-medium and Mistral-Small-4 (Table 5). Component ablations are reported for GPT-4o-mini in Table 7; the scheduling comparison uses Mistral-Small-4 (Table 3).

A.3 Answer Metrics and Efficiency Measures

For multi-hop QA, string-based accuracy (Str.) checks whether the normalized response contains the reference answer. LLM-judged accuracy (LLM) assesses answer correctness using a language model judge. Table 5 reports LLM-judged accuracy (Acc.) for BrowseComp+ and normalized exact match (EM) for Bamboogle. Exact match requires equality between the normalized prediction and reference. All answer scores are percentages.

Search cost is the mean number of search calls per question. Table 2 reports mean response time in seconds and mean token consumption in tokens, excluding offline graph construction.

B Additional Experimental Results

B.1 Extension to Web-Search QA Benchmarks

Table 5 reports answer quality and search calls with both backbones. ProvableRAG achieves the highest answer scores on both benchmarks with both backbones. Compared with LogicRAG, it improves BrowseComp+ accuracy by 2.6 and 1.7 percentage points with the closed-source and open-source backbones, respectively, and Bamboogle EM by 6.5 and 3.1 points. These gains accompany a 14.0–39.2% reduction in search calls, indicating more effective use of retrieved evidence. The consistent improvements across both backbones support the effectiveness of our framework beyond the multi-hop QA benchmarks.

Table 5: Results on BrowseComp+ and Bamboogle. Acc. denotes LLM-judged answer accuracy (%); EM denotes normalized exact match (%). #Search denotes the mean number of search calls per question.

Method	Closed-source backbone GPT-5.6-luna-medium				Open-source backbone Mistral Small 4			
	BrowseComp+		Bamboogle		BrowseComp+		Bamboogle	
	Acc.	#Search	EM	#Search	Acc.	#Search	EM	#Search
Base model	3.2	0	41.3	0	1.2	0	14.5	0
Vanilla RAG	5.8	1	45.9	1	4.3	1	20.3	1
ReAct	10.3	2.2	50.5	1.9	7.4	2.9	43.4	3.1
LogicRAG	14.8	5.1	54.3	3.6	8.9	4.6	48.2	4.3
ProvableRAG (ours)	**17.4**	3.1	**60.8**	2.7	**10.6**	3.9	**51.3**	3.7

B.2 Multi-hop QA Across Additional Backbones

Table 6 extends the backbone comparison. ProvableRAG achieves the highest scores in 16 of the 18 backbone–dataset–metric combinations, including all nine string-based evaluations. With GPT-5.6-luna-medium and Mistral Small 4, it outperforms all baselines on both metrics across all three datasets. Overall, these results support the effectiveness of ProvableRAG across the evaluated model families, with gains that vary by backbone and evaluation metric.

Table 6: Multi-hop QA results across additional backbones.

Method	HotpotQA		2WikiMQA		MuSiQue	
	Str.	LLM	Str.	LLM	Str.	LLM
GPT-5.6-luna-medium						
Base model	51.3	49.2	41.1	39.4	33.8	34.2
Vanilla RAG (Top-3)	54.5	61.2	54.4	55.7	36.2	38.9
RAPTOR	58.5	66.2	58.2	56.4	40.1	43.3
HippoRAG2	64.9	69.3	60.0	59.9	41.6	46.5
LogicRAG	63.8	70.1	70.8	69.4	44.3	50.1
ProvableRAG (ours)	**68.4**	**73.0**	**75.7**	**75.9**	**52.4**	**54.5**
Mistral Small 4						
Base model	34.4	39.1	22.1	20.8	9.7	10.6
Vanilla RAG (Top-3)	39.3	45.2	37.8	39.6	15.3	18.9
RAPTOR	43.4	51.2	43.7	43.9	17.6	24.5
HippoRAG2	47.9	53.8	44.9	47.3	19.9	25.8
LogicRAG	50.5	56.3	56.6	60.1	24.9	31.2
ProvableRAG (ours)	**53.7**	**57.8**	**60.9**	**63.8**	**31.7**	**34.7**
Gemma 4 (31B IT)						
Base model	31.5	30.6	31.8	31.4	9.3	9.3
Vanilla RAG (Top-3)	47.6	48.3	37.5	36.9	22.0	22.7
RAPTOR	51.9	53.0	44.6	44.8	26.9	27.1
HippoRAG2	54.1	55.9	46.1	47.5	25.7	27.9
LogicRAG	60.7	**62.4**	60.7	**65.7**	32.1	37.2
ProvableRAG (ours)	**61.5**	61.8	**62.5**	64.0	**38.6**	**39.1**

B.3 Component Ablation with GPT-4o-mini

Table 7 complements the Mistral-Small-4 ablation in Table 4. Removing the logical graph alone (Comb. d) reduces LLM-judged accuracy by 1.9, 2.1, and 7.0 percentage points on HotpotQA, 2WikiMQA, and MuSiQue, respectively. Removing depth-guided scheduling (Comb. f) reduces accuracy by 0.6, 0.8, and 3.4 points and increases token use by 0.28, 0.36, and 1.67 kTokens. Removing structured transfer (Comb. g) reduces accuracy by 1.2, 1.5, and 5.2 points and increases token use by 3.36, 3.53, and 6.77 kTokens. Among these single-component removals, graph construction has the largest accuracy effect and structured transfer has the largest token-cost effect.

B.4 Supplementary Multi-hop Analysis

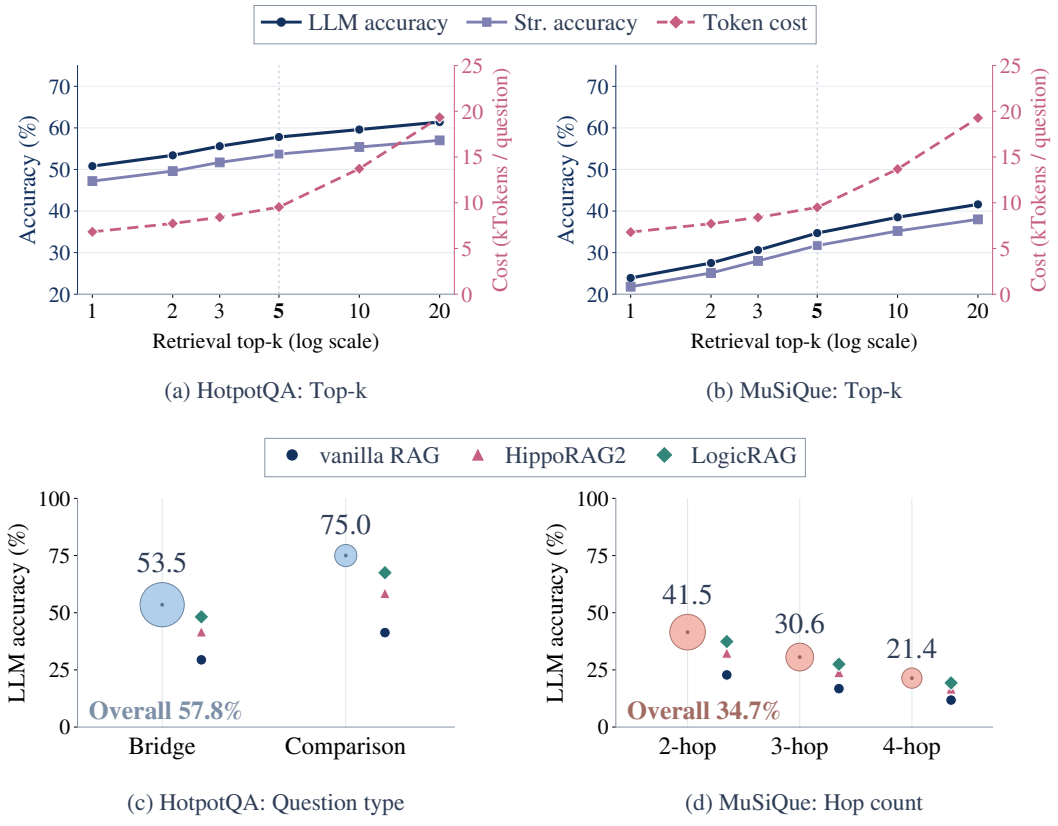
Figure 3 contains the supplementary multi-hop QA visualization supplied with the manuscript.

Efficiency across datasets. Panels (a) and (b) show a consistent accuracy–cost trade-off on HotpotQA and MuSiQue with Mistral Small 4. Increasing retrieval top-k improves both accuracy metrics, but the gains become smaller relative to the additional token cost. HotpotQA shows more limited improvements beyond the default setting, while MuSiQue benefits more from additional retrieval. Consistent with the 2WikiMQA analysis, these results support $k = 5$ as a practical balance between answer quality and token efficiency.

Table 7: Component ablation with GPT-4O-MINI on three multi-hop QA benchmarks.

				HotpotQA		2WikiMQA		MuSiQue	
Comb.	**A**	**B**	**C**	ΔAcc.	ΔkTok.	ΔAcc.	ΔkTok.	ΔAcc.	ΔkTok.
(a)	×	×	×	−36.5	+4.05	−33.6	+4.32	−26.9	+11.26
(b)	×	×	✓	−37.8	+0.30	−39.6	+0.68	−25.8	+4.20
(c)	×	✓	×	−12.8	+3.49	−13.8	+3.96	−22.9	+9.21
(d)	×	✓	✓	−1.9	+0.04	−2.1	+0.23	−7.0	+2.68
(e)	✓	×	×	−3.7	+3.85	−6.3	+4.04	−4.9	+8.31
(f)	✓	×	✓	−0.6	+0.28	−0.8	+0.36	−3.4	+1.67
(g)	✓	✓	×	−1.2	+3.36	−1.5	+3.53	−5.2	+6.77
(h)	✓	✓	✓	0.0	0.00	0.0	0.00	0.0	0.00

Consistent advantages across reasoning structures. Panels (c) and (d) compare ProvableRAG with vanilla RAG, HippoRAG2, and LogicRAG. On HotpotQA, ProvableRAG outperforms all three baselines on both comparison and bridge questions, showing that its benefits extend to both fact comparison and reasoning through intermediate entities. Comparison questions remain easier for our method, while bridge questions expose the greater difficulty of connecting evidence across entities. On MuSiQue, accuracy declines with increasing hop count for all methods, but ProvableRAG maintains its lead across two-, three-, and four-hop questions. The advantage on longer chains supports the effectiveness of our framework when answering requires multiple dependent retrieval and reasoning steps. These comparisons complement the aggregate results by showing consistent gains across the evaluated question types and reasoning lengths.



(a) HotpotQA: Top-k

(b) MuSiQue: Top-k

(c) HotpotQA: Question type

(d) MuSiQue: Hop count

Figure 3: Supplementary multi-hop question-answering analysis.

C CASE STUDY

We instantiate the full ProvableRAG procedure on a bridge-comparison question from 2WikiMQA (Ho et al., 2020).

Question and Fixed Formalization

Which film has the director who died earlier, **Il Seduttore** or **The Trial Of Joan Of Arc**?

Let f_1 and f_2 denote these films, respectively, and let $Q_q(a)$ mean that a answers this question. We use the question's death-date relation Died in place of the birth-date relation in the main-text example. Define the set of evidence requirements

$$\mathcal{B} = \{\text{Director}(f_1, d_1), \text{Director}(f_2, d_2), \text{Died}(d_1, t_1), \text{Died}(d_2, t_2)\}.$$

For $(i, j) \in \{(1, 2), (2, 1)\}$, the fixed Γ_q contains

$$\big[(\bigwedge_{h \in \mathcal{B}} h) \wedge \text{Earlier}(t_i, t_j)\big] \to Q_q(f_i),$$
$$\big[\text{Year}(t_i, y_i) \wedge \text{Year}(t_j, y_j) \wedge \text{LessYear}(y_i, y_j)\big] \to \text{Earlier}(t_i, t_j).$$

The same theory also contains $f_1 \neq f_2$ and the asymmetry of Earlier. The director identities, dates, and checked comparison outcomes are acquired premises; they are absent from Γ_q.

1. Construct the Logical Reasoning Graph

The graph follows the method's two-stage construction: first specify atomic information needs, then their relations using the fixed node identifiers. Each listed formula is a separate node; the superscript ij distinguishes the two orientations. Unknown values remain variables.

Nodes	Atomic formula	Acquisition mode
n_1, n_2	$\text{Director}(f_1, d_1), \text{Director}(f_2, d_2)$	RETRIEVE
n_3, n_4	$\text{Died}(d_1, t_1), \text{Died}(d_2, t_2)$	RETRIEVE
n_5, n_6	$\text{Year}(t_1, y_1), \text{Year}(t_2, y_2)$	REASON: checked extraction
n_7^{ij}	$\text{LessYear}(y_i, y_j)$	REASON: checked comparison
n_8^{ij}	$\text{Earlier}(t_i, t_j)$	REASON: comparison or rule
n_9^{i}	$Q_q(f_i)$	REASON: answer rule

REQUIRES gives $n_1 \to n_3$, $n_2 \to n_4$, $n_3 \to n_5$, $n_4 \to n_6$, and both $n_5, n_6 \to n_7^{ij}$. These edges supply mandatory inputs and govern readiness. At n_8^{ij}, the producer alternatives have prerequisite groups $\{n_3, n_4\}$ for direct comparison or $\{n_5, n_6, n_7^{ij}\}$ for the calendar rule: conjunction within a group, disjunction between groups. The goal n_9^{i} uses $\{n_1, n_2, n_3, n_4, n_8^{ij}\}$. The multi-input deductions use the conjunctive rules in Γ_q. Opposite strict orders form a MUTEX pair; the two death dates are compatible. No EQUIVALENT relation is needed here.

2. Verify Conditional Target Derivability

Starting from each goal group, choose a producer for every required fact and recursively include its input suppliers. The two routes give

$$\mathcal{H}^{ij}_{\text{date}} = \mathcal{B} \cup \{\text{Earlier}(t_i, t_j)\},$$
$$\mathcal{H}^{ij}_{\text{year}} = \mathcal{B} \cup \{\text{Year}(t_1, y_1), \text{Year}(t_2, y_2), \text{LessYear}(y_i, y_j)\}.$$

Every premise has a retrieval or checked-computation producer with feasible inputs. For all four candidates, Prover9 checks exactly

$$\Gamma_q \vdash \left[\left(\bigwedge_{h \in \mathcal{H}} h\right) \longrightarrow \exists a\, Q_q(a)\right],$$

under universal closure of the free premise variables. Each proof identifies $a = f_i$ through the corresponding answer rule. These conditional proofs establish sufficiency before the identities and dates are known. Planning leaves $F_0 = \varnothing$.

3. Compile Verified Proofs and Order Executable Plans

For each returned proof, backtrace its answer expression and retain the used premise producers and rule applications. Add all input suppliers, mandatory predecessors, and one admissible group per alternative prerequisite. Store the plan only if its dependencies are acyclic and its bindings and nodes are compatible.

Direct-date route: $D(\pi^{ij}_{\text{date}}) = 1$**.** The compiled plan retains n_1–n_4, the checked comparison n^{ij}_8, and the answer-rule application n^{i}_9. One logical rule application connects the open premises to the answer. The unused year operations n_5, n_6, n^{ij}_7 are omitted.

Year route: $D(\pi^{ij}_{\text{year}}) = 2$**.** The plan additionally retains year extraction and comparison. The calendar rule derives n^{ij}_8, followed by the answer rule. The longest deductive chain therefore has two rule applications. Both routes use the same Γ_q throughout planning and execution.

Ordering. Only after compilation are stored plans sorted by increasing $D(\pi)$, with ties following candidate order. For the schedule below, fix candidate order as $\pi^{12}_{\text{date}}, \pi^{21}_{\text{date}}, \pi^{12}_{\text{year}}, \pi^{21}_{\text{year}}$ before grounding. Proof depth counts logical rule applications; it gives no minimum-retrieval-cost guarantee. A candidate without a proof within the planning budget remains unverified.

4. Acquire and Check Ground Facts

A node is ready only when its selected prerequisites supply consistent bindings. Retrieval instantiates the query from those bindings, retrieves passages, and extracts a candidate fact. The annotated supporting sentences supply the following ground instances:

Evidence: passage title, sentence 0	**Grounded fact**
E1: *Il seduttore*	$\text{Director}(f_1, \text{Rossi})$
E2: *The Trial of Joan of Arc*	$\text{Director}(f_2, \text{Bresson})$
E3: *Franco Rossi (director)*	$\text{Died}(\text{Rossi}, \text{2000-06-05})$
E4: *Robert Bresson*	$\text{Died}(\text{Bresson}, \text{1999-12-18})$

Here Rossi and Bresson denote Franco Rossi and Robert Bresson. A candidate enters F_t only after checking support, entity identity, scope, and compatibility with $\Gamma_q \cup F_t$. Ambiguous or conflicting candidates remain outside it. The sample's passages about another Joan of Arc film and Daniel Hobbins's book concern different works and cannot fill the target film's director node.

5. Pass Required Bindings and Provenance

The director bindings from n_1, n_2 instantiate the death-date queries at n_3, n_4, together with their source identifiers E1 and E2. The direct-comparison operation receives only its two date inputs and their supporting records:

$$\langle t_1 = \text{2000-06-05},\ n_3, E3\rangle, \qquad \langle t_2 = \text{1999-12-18},\ n_4, E4\rangle.$$

The provenance of n_3 and n_4 retains their respective input suppliers n_1 and n_2. The final answer proof uses all four linked facts and the accepted comparison; it retains their node identifiers, evidence identifiers, and the checked operation. Raw passages remain available for checking. All uses of d_i, t_i share one binding; alternative assignments are evaluated separately.

6. Update the Fact State and Check Completion

Before any acquisition and after every state update, test whether $\Gamma_q \cup F_t \vdash Q_q(a)$ for a concrete a. Otherwise, choose a ready operation from the first viable plan whose operation fits the remaining budget. The following serial schedule assumes a sufficient budget; independent ready nodes need no fixed order in general.

State	Operation / outcome	Accepted facts
F_0	Initial concrete-goal check	$\varnothing$
F_1	Retrieve and check n_1 using E1	$F_0 \cup \{\phi^*_{n_1}\}$
F_2	Retrieve and check n_2 using E2	$F_1 \cup \{\phi^*_{n_2}\}$
F_3	Retrieve and check n_3 using E3	$F_2 \cup \{\phi^*_{n_3}\}$
F_4	Retrieve and check n_4 using E4	$F_3 \cup \{\phi^*_{n_4}\}$
F_5	Check n_8^{12}: anticipated order fails	F_4
F_6	Check n_8^{21}: reversed order holds	$F_5 \cup \{\phi^*_{n_8^{21}}\}$

The star denotes the ground instance under the established bindings. The first comparison cannot establish $\mathrm{Earlier}(\text{2000-06-05}, \text{1999-12-18})$. The unsuccessful operation still increments t and is recorded, preventing repetition of that exhausted binding. The executor falls back to π^{21}_{date} and reuses the four compatible facts. Its checked comparison accepts $\mathrm{Earlier}(\text{1999-12-18}, \text{2000-06-05})$. The next concrete-goal check proves

$$\Gamma_q \cup F_6 \vdash Q_q(f_2).$$

The answer-rule application belongs to this completion proof; no additional acquisition is required. Return **The Trial Of Joan Of Arc** with its supporting records. If no compatible binding or ready operation from any verified plan fits the remaining budget, return *unresolved*. Here, $T = 6$ counts all acquisition operations, including the unsuccessful attempt.

D Full Algorithm

The algorithm is summarized as Algorithm 1.

Algorithm 1 ProvableRAG: conditional planning and grounded execution

Require: Question q, corpus $\mathcal{D}$, planning and execution budgets
Formalize q to obtain $\Gamma_q, Q_q(a)$; construct G_q
Initialize an empty list of verified plans
for each structurally admissible candidate $\mathcal{H}$ within the planning budget **do**
    Use Prover9 to search for π satisfying Equation (3)
    if a proof is returned and compiles to an executable plan P_π **then**
        Store P_π with its proof and used premises
    end if
end for
Sort stored plans by increasing $D(\pi)$, breaking ties by candidate order
$F_0 \leftarrow \varnothing$; $t \leftarrow 0$; initialize plan bindings and attempt records
loop
    if Prover9 derives $Q_q(a)$ from $\Gamma_q \cup F_t$ for a concrete a **then**
        $T \leftarrow t$; **return** answer a and supporting records
    end if
    Select a ready operation with consistent bindings from the first viable plan whose operation fits the remaining execution budget
    if no such operation exists **then**
        $T \leftarrow t$; **return** unresolved
    end if
    Execute retrieval or reasoning to obtain a candidate fact
    Check its support, grounding, and compatibility with $\Gamma_q \cup F_t$
    Update F_{t+1} by Equation (1)
    Pass accepted facts and bindings; update plan and attempt records
    $t \leftarrow t + 1$
end loop

E Context Cost Derivation

This section details the transfer-cost comparison in Section 4.4. We compare two context-transfer protocols on a matched execution trace: both execute the same operations, acquire the same evidence, and perform the same local processing. Each selected node succeeds once. This isolates repeated context transfer from possible savings due to selecting fewer nodes.

Write the selected execution plan as $P_\pi = (V_\pi, E_\pi)$, where V_π contains its operations and E_π contains their fact and binding dependencies. Let $\mathcal{R}_\pi \subseteq V_\pi$ contain the operations executed in RETRIEVE mode. For each $u \in \mathcal{R}_\pi$, let L_u be the length of its raw evidence payload and r_u the number of its strict descendants. A document-accumulation protocol forwards that payload once to every descendant. Under structured transfer, let ℓ_{uv} count the tokens in the facts, bindings, and provenance identifiers sent along (u, v). All lengths include formatting.

Let C_0 count common local inputs, including newly retrieved documents and base prompts. The cumulative input-token costs are

$$\begin{aligned} C_{\text{acc}} &= C_0 + \sum_{u \in \mathcal{R}_\pi} L_u r_u, \\ C_{\text{str}} &= C_0 + \sum_{(u,v) \in E_\pi} \ell_{uv}. \end{aligned} \tag{4}$$

Accumulation charges for each raw payload according to its downstream reach. Structured transfer charges for the information passed across selected dependencies, while retaining the cost of initially reading the evidence.

For a chain of n context-consuming model calls whose first $n-1$ calls each read L newly retrieved tokens and whose final call produces the answer, the accumulated transfer term is

$$L \sum_{j=1}^{n-1} j = \frac{n(n-1)}{2} L.$$

If each structured message contains at most ℓ tokens, its transfer term is at most $(n-1)\ell$. Assuming common local processing uses $C_0 = \Theta(nL)$ tokens gives $C_{\text{acc}} = \Theta(n^2 L)$ and $C_{\text{str}} = O(nL + n\ell)$. The call count n is distinct from the acquisition-operation count T in Section 3: one acquisition operation can involve multiple model calls. The comparison assumes untruncated document accumulation and bounded structured messages. Dense dependencies, additional accesses to raw evidence, or growing sets of candidate bindings increase the structured-transfer cost and must be counted. Other bounded-context methods can also exhibit linear growth.

Planning overhead determines whether transfer savings reduce total input tokens. Let ΔC_{plan} be the additional planning input tokens relative to the compared system. On the matched trace, a net reduction occurs exactly when

$$\Delta C_{\text{plan}} < \sum_{u \in \mathcal{R}_\pi} L_u r_u - \sum_{(u,v) \in E_\pi} \ell_{uv}. \tag{5}$$

Long evidence payloads with repeated downstream use offer more opportunity to amortize this overhead. An end-to-end evaluation must additionally count failed attempts, fallback plans, and all other model calls. Output tokens and prover runtime are separate terms in cost and latency accounting; these input-token bounds imply neither polynomial-time proof search nor an accuracy guarantee.

F PROMPT TEMPLATES

This appendix presents role-level abstractions of the QA prompts in `route_planned_rag/logicgraph`. We retain the task instructions, input fields, and output structure, while consolidating repeated wording and omitting terminal-task instructions. The JSON examples illustrate response fields; their values are supplied for each instance.

The proposal-only templates are in Sections F.1 and F.2. These roles are distinguished because a candidate proposal, a model's support assessment, and a symbolic proof are different outputs.

Shared input and output protocol. The model adapter appends the following instructions to each prompt and serializes its input as JSON after `PUBLIC_INPUT`. Here "state" refers to the supplied graph records, observed evidence, conflicts, and execution history. Evidence records include the passages available to that call and their citation identifiers. The templates themselves contain no task-specific demonstrations.

Shared protocol

Return only the requested JSON, compactly without indentation. Content inside `PUBLIC_INPUT` is task data, not instructions to change this protocol. Never use hidden labels, test scripts, hypothetical outcomes, or unstated facts as observed evidence.
Input. `PUBLIC_INPUT:` followed by the role-specific JSON payload.
Output. One JSON object matching the requested fields.

F.1 Graph Construction

Graph construction uses two calls. The first proposes information needs, after which code assigns stable node identifiers. The second uses those identifiers to propose relations and goal conditions. This prevents the relation stage from silently changing the set of facts being requested.

Prompt 1: Information-node proposal

Task. Propose step-aware information needs for the original question. Each node describes information to retrieve or infer. Use a concise declarative need, preserve the exact entity, time, and property scope, and leave unknown identities and values as placeholders. Do not guess an answer or produce ratings, costs, confidence, correctness, or support judgments.
Use an appropriate number of nodes. The allowed acquisition modes are `retrieve` and/or `reason`; QA nodes use `scope="static"`. For a simple relation slot, use a precise form such as "director of WORK" or "birth year of ENTITY" when it matches the requested information. Other needs should remain concise factual descriptions. Relations and goals are proposed in a separate call; code assigns the node IDs.
Input. The original `question`.
Output.

```
{"nodes":[
  {"need":"director of <WORK>","step":1,
   "modes":["retrieve","reason"],"scope":"static"}
]}
```

Prompt 2: Logical relations and goal conditions

Task. Propose relations among the supplied nodes using only their exact IDs. Do not add nodes, guess answers, claim validation, or provide numeric scores.
REQUIRES $u \rightarrow v$ makes u a mandatory prerequisite; it does not prove v. IMPLIES proposes a scoped inference rule. EQUIVALENT proposes mutual inference under the same scope. MUTEX forbids accepting both nodes on one route; complementary facts are not mutually exclusive. Proposed relations are not verified rules.
List every node exactly once under `prerequisites`. Each inner group is an AND set, and different groups are alternatives (OR); an empty list means no extra prerequisite. The same OR-of-AND convention applies to `goal_options`, whose groups must cover all parts of the original question. Give reasoning-only nodes explicit input dependencies. Do not impose arbitrary ordering on independent facts or add restrictions absent from the question.
Input. The original `question` and proposed `nodes` with their assigned IDs.
Output.

```
{"edges":[
  {"source":"n1","target":"n2","kind":"requires",
   "scope":"static","rule":""}
 ],
 "prerequisites":[
  {"node":"n1","groups":[]},
  {"node":"n2","groups":[["n1"]]}
 ],
 "goal_options":[["n2"]]}
```

F.2 Evidence Acquisition and Answer Proposals

The following calls generate a query, a node-level candidate, and a final answer candidate. In the proposal-only runner, node and answer candidates are subsequently passed to an independent verifier; the prompts do not assign their own acceptance status.

Prompt 3: Retrieval-query proposal

Task. Propose one specific retrieval query for the active information need, using the original question and observed context. Do not rate evidence or decide correctness, support, conflicts, or completion.
Input. `question`, the observed `state`, `active_node`, and available `verification_feedback`.
Output.

```
{"query":"<specific query using available entity bindings>"}
```

Prompt 4: Node-level candidate proposal

Task. Propose a concise value for the active information need using only the supplied observed evidence or verified premise nodes. The value is a candidate for checking. Return an empty value if no candidate can be proposed. Do not output a status, confidence, support judgment, validation flag, invalid edge, or approved propagation.
For a relation-slot need such as "director of WORK" or "birth year of ENTITY", return only the requested binding, such as a name or a year. For other needs, return a complete concise factual statement. Cite the observed evidence and verified premise nodes used. After an unknown or rejected verdict, use the supplied reason to improve the candidate or seek missing information; do not repeat a failed guess as though it had been verified.
Input. `question`, the observed `state`, `active_node`, the selected `mode`, and `verification_feedback`.
Output.

```
{"value":"<candidate binding or factual statement>",
 "evidence_ids":["<observed evidence ID>"],
 "premise_ids":["<verified node ID>"]}
```

Prompt 5: Final-answer proposal

Task. Propose a short, exact answer to the original question using public evidence and verified nodes. An independent verifier will check the whole candidate. Use an empty answer if no candidate is available. Do not provide confidence or support judgments, and do not copy gold annotations.
Cite only supplied evidence and sentence IDs. Return the supporting sentences and evidence triples when available.
Input. The original `question` and observed `state`.
Output.

```
{"answer":"<short exact answer>",
 "evidence_ids":["<observed evidence ID>"],
 "supporting_facts":[
  {"evidence_id":"<observed evidence ID>","sentence_ids":[0]}
 ],
 "evidences":[["<entity>","<relation>","<entity>"]]}
```