GPS: GRAPH-GUIDED PROACTIVE INFORMATION SEEKING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Equipping Large Language Models (LLMs) with the ability to proactively ask clarifying questions is essential to mitigate ambiguity when faced with underspecified user queries in retrieval-augmented generation (RAG) systems. However, existing methods often neglect the rule-based reasoning structures embedded in the retrieved knowledge that are central to ambiguity, making it challenging to learn an effective and efficient question-asking strategy. To address these issues, we introduce **GPS**, a two-stage framework for enhancing proactive information seeking abilities of LLMs in RAG systems. In the reasoning stage, we propose a Directed Acyclic Graph (DAG) reasoning structure with theoretical guarantees of logical completeness, which facilitates capturing all conditional logic in the retrieved knowledge and supports effective clarification. In the clarification stage, we design a traversal-based algorithm that dynamically prunes the DAG based on user responses, enabling efficient clarification. To further enhance DAG construction, we first propose a conditional paths guided data synthesis method to address data scarcity challenge, then we apply a clarification-oriented reinforcement learning method with a hybrid reward that jointly considers effectiveness and efficiency to optimize the LLM. Experiments on three benchmarks demonstrate that **GPS** outperforms baseline methods in both success rate and interaction cost.

1 INTRODUCTION

Consider a user seeking information about disability benefits eligibility and asks a question: “Am I eligible for disability premium?” While this question seems straightforward, the actual eligibility depends on multiple unstated conditions: income level, disability severity, and age. Without this critical information, even the most advanced retrieval-augmented generation (RAG) systems may provide incorrect or misleading answers. This scenario, illustrated in Figure 1, exemplifies a fundamental challenge in real-world question-answering systems: *how can AI systems proactively identify and gather missing information to provide accurate responses?*

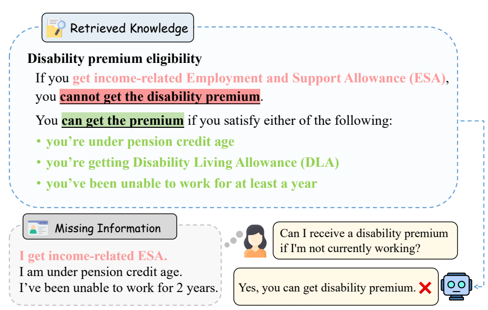

Figure 1: An illustration of user underspecified queries. The user asks information about disability benefits eligibility. However, when user queries lack sufficient information such as income level, disability severity and age, LLMs may generate incorrect responses.

The ambiguity stems from underspecified user queries, which are common in real-world settings due to users' limited domain knowledge (Zhang et al., 2024; Deng et al., 2023a; Kim et al., 2024) or natural tendency to omit seemingly obvious details (Zipf, 1949). While existing RAG methods excel at retrieving relevant documents (Ren et al., 2025; Lewis et al., 2020; Zhao et al., 2025), they *fundamentally assume that user queries contain sufficient information, but this assumption often fails in practice*.

A promising solution is to equip Large Language Models (LLMs) in RAG systems with the ability to proactively ask clarifying questions when faced with underspecified queries. Currently, there are two main approaches: prompting and fine-tuning. Prompting methods (Deng et al., 2023b; Kuhn et al., 2023; Hu et al., 2024; Kobalczyk et al., 2025) utilize the reasoning capabilities of LLMs to iteratively identify ambiguity and generate clarification questions. However, their performance is constrained by the capability of LLMs as small-scale LLMs often struggle to identify ambiguities (Zhang et al., 2024). The better way is to fine-tune LLMs by multi-turn clarification dialogue data collected through human annotation (Qian et al., 2024; Chen et al., 2024) or self-sampling strategies (Andukuri et al., 2024; Zhang et al., 2025). However, the former is costly to obtain, while the latter imposes no constraints on the clarification search space, potentially leading to irrelevant or redundant interactions. Therefore, it is necessary to develop an **effective** and **efficient** method to reach our goal.

We propose that the key to resolving ambiguity in underspecified queries lies in explicitly modeling the conditional reasoning structures within retrieved documents. Unlike existing methods that treat clarification as an open-ended dialogue problem, we observe that domain-specific documents typically encode knowledge as conditional rules—if-then statements that map combinations of conditions to conclusions. By extracting and representing these rules as a Directed Acyclic Graph (DAG), we can systematically identify all conditions relevant to the user's query and guide clarification dialogues through efficient traversal strategies.

However, realizing this vision presents three fundamental challenges. **(C1) How can we design a reasoning structure that captures all logical dependencies while remaining computationally tractable?** The structure must be expressive enough to represent arbitrary Boolean functions yet efficient enough for real-time interaction. **(C2) How can we train models to extract such structures when existing datasets lack annotations for conditional reasoning?** Current QA benchmarks rarely include underspecified queries or their missing conditions. **(C3) How can we optimize the extracted structures for both correctness and interaction efficiency?** Users will abandon systems that require excessive clarification rounds.

To address **(C1)**, we propose a conditional reasoning DAG structure, which is theoretically guaranteed to be logically complete to express any Boolean function via disjunctive normal form (DNF). Besides, the DAG allows for subgraph sharing across reasoning paths and supports dynamic pruning based on user responses, enabling $O(r)$ average-case clarification complexity, where $r \ll k$ is the average reasoning depth rather than the total number of conditions k. To address **(C2)**, we propose a conditional path guided data synthesis method to generate usable dataset for both training and evaluation. This method generates question-answer pairs with associated missing conditions along each conditional path from document. A filtering mechanism based on the necessity of the missing conditions is further applied to retain high-quality examples. To address **(C3)**, We propose a clarification-oriented reinforcement learning method to enhance LLM's ability to extract DAG structures for effective and efficient clarification. We design a hybrid reward that encourages the LLM to prioritize DAG that leads to correct answer and requires fewer interaction.

Our main contributions can be summarized as follows:

- **Novel Framework**: We introduce **GPS** (**G**raph-guided **P**roactive Information **S**eeking), the first framework to explicitly model conditional reasoning structures for clarification in RAG systems.
- **Theoretical Foundation**: We prove that our DAG-based representation achieves logical completeness while enabling $O(r)$ average-case clarification complexity, where $r \ll k$ is the average reasoning depth rather than the total number of conditions k.
- **Practical System**: We develop a complete pipeline including (i) Conditional path guided synthetic data generation to address training data scarcity, (ii) clarification-oriented reinforcement learning that jointly optimizes for accuracy and efficiency, and (iii) dynamic traversal algorithms that reduce user interaction burden.
- **Empirical Validation**: Extensive experiments on three benchmarks demonstrate that **GPS** achieves average improvement of **7.5%** in success rate and **4.2%** in clarification efficiency over the best baseline method.

2 Related Work

Clarification in LLMs Currently, there are two main approaches to enhance the ability of LLMs to proactively ask clarifying questions: prompting and fine-tuning. Prompting methods (Deng et al., 2023b; Kuhn et al., 2023) utilize the reasoning capabilities of LLMs to iteratively identify ambiguity based on the conversation history and choose to either ask clarification questions or generate response. However, their performance is constrained by the capability of LLMs as small-scale LLMs often struggle to identify ambiguities (Zhang et al., 2024), and as the conversation history grows longer, the risk of lost-in-the-middle increases (Liu et al., 2024). Another line of work is to fine-tune LLMs with multi-round conversation data (Qian et al., 2024; Chen et al., 2024). Yet these approaches rely on access to human-annotated conversation data, which is expensive to collect in practice. Some methods (Andukuri et al., 2024; Zhang et al., 2025) explore self-improve paradigm for sampling conversation data and use the accuracy of final responses to filter low-quality clarification data. Nevertheless, these methods typically imposes no constraints on the clarification search space, potentially leading to irrelevant or redundant interactions. Therefore, it is necessary to develop an **effective** and **efficient** method for proactive clarification.

Graph-based Reasoning in NLP Recent work has explored structured representations for multi-hop reasoning (Besta et al., 2024), knowledge graph integration (Ren et al., 2020; Li et al., 2025), and neural-symbolic reasoning (Xu et al., 2024). Query2Box (Ren et al., 2020) reasons over knowledge graphs by embedding multi-hop logical queries as geometric boxes in vector space. Li et al. (2025) proposes to inject LLMs with structured knowledge by encoding knowledge graphs via graph neural networks(GNNs). Xia et al. (2025) proposes a novel fine-tune framework stimulating the ability of LLMs to perform complex reasoning on knowledge graphs. However, these methods focus on reasoning over existing knowledge rather than proactive information seeking. Our work uniquely combines graph-based reasoning with interactive clarification.

3 Problem Formulation

In this paper, we aim to enhance LLMs' ability to proactively ask clarification questions when facing underspecified user query in RAG scenarios. Rigorously, given a user query q, the retrieved relevant document $d = Retrieve(q)$, and the user's background context S which is not observable to the LLM, we denote by $C_d = \{c_1, \ldots, c_k\}$ the set of user-specific condition variables in d, each condition variable c_i takes values from a finite value set $\mathcal{V}_{c_i}$.[1] We divide C_d into two disjoint subsets:

- $C_{\text{known}}(q) \subseteq C_d$: the subset of **known condition variables** with values provided in query q.
- $C_{\text{miss}}(q) = C_d \setminus C_{\text{known}}(q)$: the subset of **missing condition variables** specific to query q, with values depend on the hidden user's background S and are necessary to determine the answer.

Let $A = \{a_1, \ldots, a_m\}$ denote the set of possible answers. The final answer $a \in A$ is determined by $C_{\text{miss}}(q)$ through a set of latent logical constraints $\mathcal{R}$ encoded in d (e.g., eligibility rules). Our objective is to enhance LLMs' ability to proactively elicit the values of $C_{\text{miss}}(q)$, so that an unambiguous answer a can be inferred.

4 Methodology

In this section, we introduce **GPS**, a two-stage framework for proactive clarification. In the reasoning stage, a Reasoner LLM Θ_R captures the conditional structure in documents as a DAG. In the clarification stage, a Clarifier LLM Θ_C interacts with a User-Simulator LLM Θ_U, dynamically pruning the DAG during traversal to elicit values of $C_{\text{miss}}(q)$. To further improve DAG quality, **GPS** employs the Conditional Path Guided Data Synthesis procedure to construct training dataset and utilizes Clarification-oriented Reinforcement Learning to optimize the Reasoner. The overall pipeline is illustrated in Figure 2.

4.1 Conditional reasoning DAG construction

To construct a structure which is both logically complete and clarification-efficient, we define a conditional reasoning DAG structure $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where:

- Each non-terminal node $n_{c_i} \in \mathcal{N}$ represents a user condition variable $c_i \in C_d$, each terminal node $n_{a_m} \in \mathcal{N}$ represents a possible answer $a_m \in A$.

[1] For example, a condition variable $c_i =$ "marital status" may have $\mathcal{V}_{c_i} = \{\text{Single}, \text{Married}, \text{Divorced}\}$.

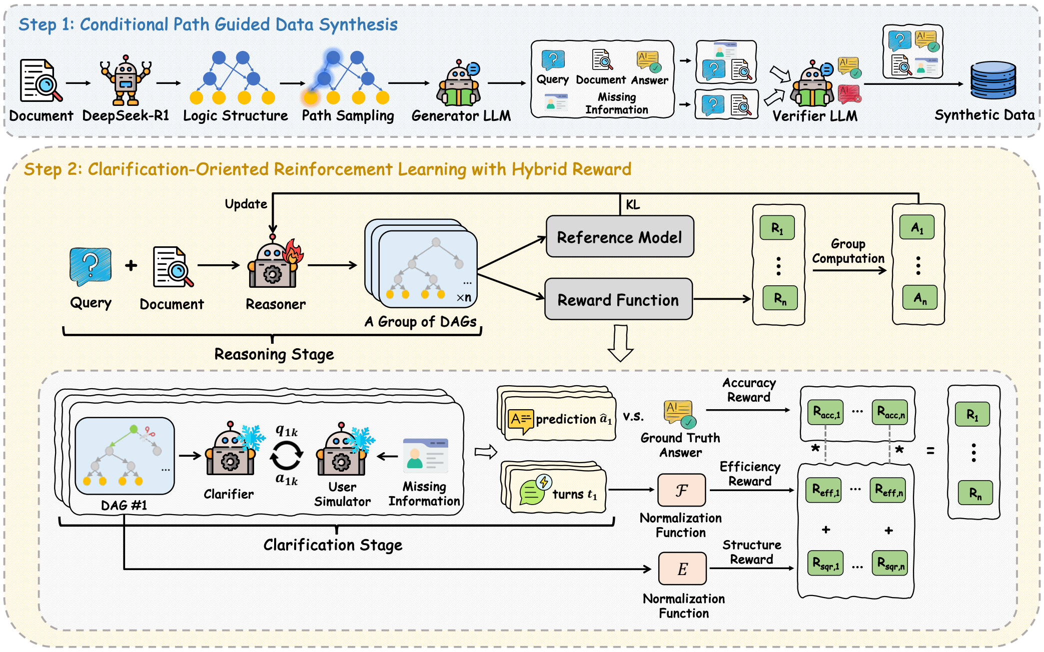

Figure 2: The overview of our method pipeline.

- Each edge $e_{i,j} = (n_{c_i}, n_{c_j}, \nu)$ is labeled with a condition value $\nu \in \mathcal{V}_{c_i}$, denoting a possible value of the predecessor node. The outgoing edges from each node are mutually exclusive and collectively exhaustive.
- If a node has a single predecessor node, it implicitly forms an **AND** relation with its predecessor condition. If a node has multiple predecessor nodes, it forms an **OR** relation over all predecessor conditions.

We first analyze the logical completeness of our proposed conditional reasoning DAG structure as follows. The clarification efficiency will be discussed in 4.2.

Proposition 1. *For any finite-valued function $g : \prod_{i=1}^{k} \mathcal{V}_i \to A$ over condition variables $\{c_i\}_{i=1}^{k}$, there exists a conditional reasoning DAG $\mathcal{G}$ such that, for each $a_m \in A$, every root-to-leaf path ending at a_m corresponds to a conjunction in the disjunctive normal form (DNF) of the indicator function $\mathbf{1}[g(\cdot) = a_m]$, and the union of all such paths encodes the full DNF of $\mathbf{1}[g(\cdot) = a_m]$.*

See Appendix A for a complete proof. The proposition theoretically guarantees that our conditional reasoning DAG structure is expressive enough to represent any finite-valued function, allowing for comprehensive detection of missing conditions and effective clarification.

In the reasoning stage, we prompt the Reasoner LLM Θ_R to construct a conditional reasoning DAG based on the user query and the retrieved document. The detailed prompt is provided in Appendix G.1.

4.2 Dynamic traversal-based clarification

Given the constructed conditional reasoning DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, we propose a dynamic traversal-based clarification approach that generates clarification questions in a topological order to prioritize essential conditions and dynamically prune inconsistent paths. Formally, let $\deg_{\text{in}}(n_i)$ denote the in-degree of node n_i, we define the candidate clarification set U:

$$\begin{aligned} U_1 &= \{n_i \in \mathcal{N} \mid \deg_{\text{in}}(n_i) = 0,\ i \notin C_{\text{known}}(q)\}, \\ U_2 &= \{n_i \in \mathcal{N} \mid \exists (n_u, n_i, \nu) \in \mathcal{E},\ u \in C_{\text{known}}(q),\ \deg_{\text{in}}(n_u) = 0\}, \\ U &= U_1 \cup U_2 \end{aligned} \tag{1}$$

Nodes in U are eligible for clarification as they are not blocked by unresolved predecessors. To determine the initial order of the candidate set, we estimate the expected cost based on the remaining depth for each $n_i \in U$:

$$\ell(n_i) = \frac{1}{|P_{n_i}|} \sum_{p \in P_{n_i}} \text{len}(p) \tag{2}$$

where P_{n_i} is a set of paths from n_i to leaf node, and $\text{len}(p)$ is the number of missing condition variables in path p.

Our dynamic traversal-based clarification consists of three steps:

1. **Initialization:** construct the candidate set of missing conditions and start with empty dialogue history H;
2. **Iterative clarification:** select the most informative condition, ask a clarification question via Θ_C, obtain a user-specific answer from Θ_U, and update H by following the consistent path;
3. **Final answering:** once a terminal condition is reached or no candidates remain, Θ_C produces the final answer $\hat{a}$ conditioned on H.

Through dynamic traversal-based clarification process, the dialogue history H constitutes the elicited values for the missing condition set C_{miss}, which are required to resolve the ambiguity. We present the algorithmic pseudo-code of dynamic traversal-based clarification in Appendix C. We also analyze the efficiency of GPS and obtain that the expected number of clarification turns depends only on the small set of conditions along the true reasoning path, rather than the full set of conditions present in the document. Detailed analysis is provided in the Appendix B.

4.3 Clarification-oriented Reinforcement Learning

Enhancing the Reasoner LLM's ability to construct accurate conditional reasoning DAG is essential for the overall performance of the **GPS** framework. To achieve this, we first propose a *Conditional Path Guided Data Synthesis* procedure to address the data scarcity challenge. Based on the synthetic dataset, we design a *Clarification-Oriented Reinforcement Learning* approach with a hybrid reward that integrates clarification effectiveness and efficiency, encouraging the Reasoner LLM to extract DAG structures that lead to correct answer and require fewer interaction.

4.3.1 Conditional path guided data synthesis

We synthesize our dataset based on ConditionalQA (Sun et al., 2022), a reading comprehension dataset that includes long-context documents containing complex logic rules, along with well-specified and underspecified queries with human-annotated missing conditions. However, a key limitation of ConditionalQA dataset is that only 550 out of 2,247 samples (24.5%) are underspecified, making it difficult to train models that generalize well in proactive information seeking task. To address this, we propose *Conditional Path Guided Data Synthesis* method to augment high-quality underspecified training data. The synthesis process consists of two steps: **Problem Generation** and **Verification**.

Problem Generation First, we prompt advanced LLMs such as DeepSeek-R1 (DeepSeek-AI et al., 2025) to generate underspecified questions with multi-conditional reasoning paths from a document d. The prompt used for this task can be found in Appendix G.2. Each item contains three parts:

- An **underspecified question** q that admits multiple plausible answers;
- A set of **missing conditions** C with value domains $\{\mathcal{V}_c\}_{c \in C}$;
- A set of **conditional paths** $\mathcal{P} = \{(\mathbf{v}, a)\}$, where $\mathbf{v} \in \prod_{c \in C} \mathcal{V}_c$ is a complete assignment, $a \in A$ is the *unique* answer determined by $\mathbf{v}$.

Verification Each synthetic data instance is represented as $(q_i, \mathbf{v}_i, d_i, a_i)$. To ensure data quality, we introduce a filtering mechanism based on the necessity of the missing conditions. For each instance, we prompt a Verifier LLM to predict the answer both with and without access to the missing conditions. We retain the instance only if the full-input prediction a_{full} matches the gold answer a_i, while the masked-input prediction a_{partial} does not. This filtering preserves cases where missing information is essential, yielding a high-quality dataset $\mathcal{D} = \{q_i, \mathbf{v}_i, d_i, a_i\}_{i=1}^{n}$.

4.3.2 HYBRID REWARD: EFFECTIVENESS, EFFICIENCY, AND ENTROPY

Setup. Our goal is to train the *Reasoner* LLM Θ_R to generate high-quality conditional reasoning DAGs that enable a *fixed* Clarifier LLM Θ_C to complete proactive clarification with minimal interaction. Motivated by recent progress of RL for reasoning(DeepSeek-AI et al., 2025), we formulate the DAG-construction problem as a reinforcement learning task: the **policy** $\pi_\theta(o \mid q, d)$ autoregressively generates and parses a DAG o given query q and document d (we write θ as the Reasoner parameters Θ_R for brevity), the DAG deterministically induces a clarification trajectory (which condition to ask next and which edge to follow), and yielding a scalar **reward**. We adopt GRPO (DeepSeek-AI et al., 2025) algorithm to optimize π_θ.

GRPO objective. For each query q and retrieved document d, GRPO samples h DAGs $\{o_i\}_{i=1}^{h} \sim \pi_{\theta_{\text{old}}}(\cdot|q,d)$ with rewards $\{r_i\}_{i=1}^{h}$, defines the advantage A_i as

$$A_i = \frac{r_i - \text{mean}(\{r_1,\ldots,r_h\})}{\text{std}(\{r_1,\ldots,r_h\})}, \tag{3}$$

and optimizes the clipped policy objective with a reference KL:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\Theta) = \mathbb{E}_{q\sim D,\ \{o_i\}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \frac{1}{h}\sum_{i=1}^{h}\Big[\min\Big(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i,\ \text{clip}\big(\tfrac{\pi_\theta}{\pi_{\theta_{\text{old}}}},\ 1-\epsilon,\ 1+\epsilon\big)A_i\Big) \\ -\ \beta\,\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\Big],\end{aligned} \tag{4}$$

with

$$\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - 1. \tag{5}$$

GRPO calculates *group-relative* advantages within h samples $\{o_i\}_{i=1}^{h}$ for the same input, so higher-reward generated DAG samples are upweighted and lower-reward ones are downweighted.

Hybrid reward function. To optimize the policy toward high-quality DAGs, we design a hybrid reward function that jointly promotes correctness, low interaction cost, and structurally non-redundant.

- **Effectiveness reward** encourages correct final answers after clarification. Let $\hat{a}_i$ be the predicted answer after clarification based on DAG o_i and a_i the ground truth:

$$r_{\text{acc},i} = \begin{cases} 1, & \texttt{Evaluator}(\hat{a}_i, a_i) = \texttt{True}, \\ 0, & \texttt{Evaluator}(\hat{a}_i, a_i) = \texttt{False}. \end{cases} \tag{6}$$

- **Efficiency reward** encourages fewer clarification turns. Let t_i be the number of turns induced by o_i, and $t_{\max} = \max_{j\in[1,h]} t_j$ within the GRPO group. We set the coefficient $\alpha = 0.5$:

$$r_{\text{eff},i} = 1 - \alpha\,\frac{t_i}{t_{\max}}, \tag{7}$$

- **Structural quality reward** $r_{\eta,i}$ measures how effectively a DAG converts intermediate branching uncertainty into discriminative power over the final conclusions. We formalize this intuition using an information–conversion efficiency measure.

  Forward probability propagation. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a clarification DAG with condition nodes C and conclusion (leaf) nodes L. We assume *uniform branching*: at any condition node, the outgoing probability mass is evenly split among all children. Let $R \subseteq C$ be condition roots with no predecessors. Each root receives initial mass $1/|R|$. For any condition node $n \in C$, its outgoing mass splits uniformly across children:

$$P(v) = \sum_{u:(u\to v)\in\mathcal{E}} \frac{P(u)}{|F(u)|}, \tag{8}$$

  where $F(u)$ is the set of children of u.
  This yields a forward-reachability distribution $P(n)$ over all nodes. We can obtain the mass $P(\ell)$ of any leaf node $\ell \in L$ and the normalized leaf distribution is:

$$\tilde{P}(\ell) = \frac{P(\ell)}{\sum_{\ell'\in L} P(\ell')}. \tag{9}$$

Entropy of graph splits and leaf. The *graph split entropy* reflects the total uncertainty injected by intermediate splits:

$$H_{\text{graph}} = \sum_{n \in C} P(n) \log |F(n)|, \tag{10}$$

and the *leaf entropy* is the Shannon entropy of the normalized leaf distribution:

$$H_{\text{leaf}} = -\sum_{\ell \in L} \tilde{P}(\ell) \log \tilde{P}(\ell). \tag{11}$$

Information-conversion efficiency. We define the structural quality reward $r_{\eta,i}$ as

$$r_{\eta,i} = \begin{cases} \dfrac{H_{\text{leaf}}}{H_{\text{graph}}}, & H_{\text{graph}} > 0, \\ 1, & H_{\text{graph}} = 0 \ \wedge \ \exists \ell \in L : \ P(\ell) > 0, \\ 0, & \text{otherwise}. \end{cases} \tag{12}$$

We provide illustrative instances in Appendix H.3, demonstrating the rationality of the structural quality reward.

- **Overall reward.** The i_{th} sample's total reward is calculated as follows:

$$r_i = r_{acc_i} \cdot (r_{eff_i} + r_{\eta,i}) \tag{13}$$

We provide the pseudo-code of overall inference procedure of GPS in Appendix C and training procedure of the Reasoner in Appendix C, and additionally offer an illustrative example of the two-stage proactive information-seeking process in Appendix I.

5 Experiment

5.1 Experimental Setup

Dataset We constructed the GPS training dataset based on ConditionalQA (Sun et al., 2022) and we test our method on the following three datasets. The detailed statistics of the datasets are introduced in Appendix L.

- **Synthetic** is the test split of our conditional path guided synthetic dataset, consisting entirely of underspecified queries.
- **ConditionalQA** (Sun et al., 2022) includes both well-specified queries and underspecified queries. It provides annotation for each question-answer pair along with the document and the corresponding missing conditions. For well-specified queries, the missing conditions are empty.
- **ShARC** (Verma et al., 2020) is a conversational QA dataset that also includes well-specified queries and underspecified queries based on rules expressed in natural language text. For underspecified queries, it provides annotated clarification dialogues. We concatenate the clarification dialogue as the missing conditions. Compared to typical datasets in RAG scenarios, ShARC features much shorter documents and a restricted answer space limited to *yes* or *no*. We use ShARC to evaluate the generalization ability of our method.

Baselines We adopt the following state-of-the-art approaches as our compared baselines.

- **Base Method** answers user query directly based on the relevant document, which can be considered as fundamental framework in RAG.
- **ProCoT** (Deng et al., 2023b) is a prompt based method. It leverages a Chain of Thought prompting scheme to judge whether the user query is underspecified and generate a clarification question if needed.
- **UoT** (Hu et al., 2024) proposes Uncertainty of Thought prompting, which enhances llm reasoning by explicitly modeling and reducing uncertainty during the reasoning process.

- **BED-LLM** (Kobalczyk et al., 2025) uses Bayesian Experimental Design to pick the question that maximizes information gain, replacing implicit LLM reasoning with explicit sampling-based utility.
- **Adaptive-BED-LLM** is the ambiguity-adaptive variant of BED-LLM. We first generates a group of initial answers and prompt an evaluator LLM to judge whether these answers are semantically consistent. If consistent, the model answers directly; otherwise it proceeds with BED-based question selection. The evaluation prompt is provided in Appendix G.3.
- **Clarify-DPO** (Zhang et al., 2025) is a fine-tuning based method. It leverages a self-improve method to collect training data and filter data by gold answer.
- **Adaptive-Clarify-DPO** extends Clarify-DPO with an ambiguity-adaptive mechanism based on the *Clarify-or-Direct Answer* strategy proposed in (Zhang et al., 2025). The model learns to choose either generating a clarification question or directly answering.

Models We evaluate the performance using Llama3-8B-Instruct (Grattafiori et al., 2024) and Qwen2.5-7B-Instruct (Qwen et al., 2025) as backbone models.

Evaluation Metrics We evaluate the model's proactive information seeking ability using the following four metrics:

- **Success Rate (SR)**. Following previous studies(Hu et al., 2024; Qian et al., 2024), we use this metric measures the **effectiveness** of clarification process by computing the proportion of the correct predictions after clarification. We employ an evaluator LLM to judge the semantic alignment between the predicted answer and the ground-truth answer. The evaluation prompt we use is provided in Appendix G.
- **Mean Clarification Turns (MCT)**. This metric measures the efficiency of clarification process by computing the average number of clarification questions asked before generating the predicted answer (Hu et al., 2024; Qian et al., 2024).
- **Weighted Clarification Turns (WCT).** The desired behavior of a proactive information seeking model is to prioritize correct clarification before optimizing efficiency, while MCT alone cannot capture **success-conditioned efficiency**. Inspired by prior evaluation protocols (Yokoyama et al., 2021), we introduce the Weighted Clarification Turns (WCT):

$$\mathrm{WCT} = p_{\mathrm{success}} \cdot \mathrm{MCT}_{\mathrm{success}} + p_{\mathrm{failed}} \cdot T_{\max}, \tag{14}$$

  where p_{success} and p_{failed} denote the proportions of successful and failed samples, $\mathrm{MCT}_{\mathrm{success}}$ is the mean clarification turns over successful samples, and $T_{\max} = 10$ is the maximum clarification-turn budget in our experiment. Lower WCT reflects more efficient clarification while preserving success rate.
- **F1 score** for Clarification Need Prediction Accuracy (CNP). Following previous studies (Deng et al., 2023b; Zhang et al., 2025), we compute the F1 score of CNP for evaluating the model's ability to identify the necessity of clarification.

5.2 Performance Comparison

Table 1 presents the performance comparison of different methods across three benchmarks: Synthetic, ConditionalQA, and ShARC. We summarize key findings below:

Training for proactive information seeking is essential. The Base Method yields low success rates (SR) on the Synthetic and ShARC datasets, where the proportion of underspecified queries is substantially higher than in ConditionalQA dataset. This suggests that the Base Method struggles to handle underspecified queries. Baseline methods equipped with proactive clarification consistently improve SR over the Base Method. However, purely prompt-based methods sometimes fail to surpass the Base Method. For example, **ProCoT**, which introduces proactive clarification at the prompting level, occasionally results in degraded performance. This degradation is likely due to the limited capacity of backbone models and the inherent complexity of conditional reasoning required by the documents, consistent with observations reported by (Zhang et al., 2024).

GPS achieves the best balance between effectiveness and efficiency. Compared to existing baselines, **GPS** consistently improves SR. With LLaMA-3-8B-Instruct as the backbone, **GPS** achieves

Table 1: **Performance comparison on three datasets.** Columns report SR (Success Rate, %), Mean Clarification Turns(MCT), WCT (Weighted Clarification Turns), and F1 score (%). **Bold** indicates the best result, while underline denotes the second-best results.

Method	Synthetic SR (↑)	Synthetic MCT	Synthetic WCT (↓)	Synthetic F1 (↑)	ConditionalQA SR (↑)	ConditionalQA MCT	ConditionalQA WCT (↓)	ConditionalQA F1 (↑)	ShARC SR (↑)	ShARC MCT	ShARC WCT (↓)	ShARC F1 (↑)
Qwen2.5-7B-Instruct												
Base Method	21.2	0.0	7.88	0.0	70.3	0.0	2.98	0.0	49.3	0.0	5.08	0.0
ProCoT	42.5	0.43	6.07	50.9	71.6	0.33	2.95	10.4	62.6	0.67	4.06	51.3
UoT	32.8	1.10	7.05	89.2	60.3	0.56	4.25	28.2	70.5	0.52	3.25	83.8
BED-LLM	40.9	1.45	6.41	**100.0**	52.8	1.22	5.26	**37.6**	62.2	0.80	4.22	66.7
Adaptive-BED-LLM	34.6	1.26	6.89	95.2	50.2	0.93	5.58	28.6	59.4	0.90	4.56	71.0
Clarify-DPO	59.2	1.0	4.67	**100.0**	72.0	1.0	3.52	**37.6**	78.5	1.0	2.93	66.7
Adaptive-Clarify-DPO	32.6	0.94	7.04	96.2	69.9	0.02	3.02	0.0	70.0	0.01	2.99	0.0
GPS	**60.2**	1.35	**4.59**	96.4	**73.4**	0.78	**2.91**	36.7	**79.3**	0.89	**2.41**	**87.5**
LLaMA3-8B-Instruct												
Base Method	30.8	0.0	6.92	0.0	62.8	0.0	3.72	0.0	56.6	0.0	4.34	0.0
ProCoT	28.3	0.26	7.62	29.2	66.3	0.36	3.58	25.7	53.7	0.86	5.16	35.6
UoT	29.7	1.20	7.36	90.9	64.6	0.90	4.03	31.7	68.3	0.76	3.67	69.9
BED-LLM	39.6	1.50	6.53	**100.0**	47.2	1.50	6.02	**37.6**	64.0	0.98	4.20	66.7
Adaptive-BED-LLM	35.6	1.36	6.84	96.2	44.5	1.03	5.90	31.4	67.8	0.46	3.49	67.0
Clarify-DPO	53.2	1.0	5.21	**100.0**	66.3	1.0	4.03	**37.6**	**82.7**	1.0	**2.55**	66.7
Adaptive-Clarify-DPO	31.3	0.91	7.16	95.5	67.7	0.02	3.23	0.0	68.8	0.0	3.12	0.0
GPS	**56.5**	1.12	**5.02**	96.2	**74.6**	0.81	**2.89**	28.0	75.8	0.58	2.79	**82.5**

an average relative improvement of 10.4% over the second-best method on SR across three datasets. When using Qwen2.5-7B-Instruct as the backbone, **GPS** also yields an average relative improvement of 4.5% over the corresponding second-best SR. Importantly, when considering **WCT**, which jointly reflects success rate and efficiency, **GPS achieves the lowest WCT in nearly all settings**, demonstrating that the model achieves higher correctness with lower effective clarification cost.

Strong generalization to ShARC. **GPS** also generalizes well to the ShARC dataset, where it consistently outperforms prompt-based methods and Base Method, and achieves performance comparable to the **Clarify-DPO** method, despite the fact that **Clarify-DPO** is trained directly on ShARC. This highlights the strong generalization ability of **GPS** across different benchmarks.

5.3 Ablation Study

To evaluate the contribution of each component in **GPS**, we conduct an ablation study using Qwen2.5-7B-Instruct as the backbone LLM. Results are reported in Table 2 on both the Synthetic and ConditionalQA datasets.

The full **GPS** method achieves the best performance across all metrics. Removing the reinforcement learning objective (**w/o RL**) leads to a clear drop in SR on the **Synthetic** dataset from 60.2 to 52.2, confirming the effectiveness of policy optimization. Ablating the **Efficient Reward** and **Structural quality Reward** design both harms performance, decreasing SR and increasing both WCT and MCT on **Synthetic** dataset, indicating the necessity of jointly modeling correctness and efficiency. Finally, disabling the dynamic traversal mechanism (**w/o Dynamic Traversal**) leads to performance degradation on both datasets, especially increased MCT from 1.35 to 1.84 on **Synthetic** dataset and from 0.78 to 0.86 on **ConditionalQA** dataset, suggesting its role in optimizing clarification paths.

5.4 Qualitative Analysis

Figure 3 presents comparison between a strong baseline Clarify-DPO and our method GPS on an underspecified query about policy eligibility. Clarify-DPO selects clarification questions based on implicit reasoning, which leads to missing essential condition about income-related ESA, and the resulting clarification path does not cover all necessary branches. This omission causes the model to produce an incorrect final answer.

Table 2: Ablation results of GPS with Qwen2.5-7B-Instruct.

Method	Synthetic				CondQA			
	SR↑	MCT	WCT↓	F1↑	SR↑	MCT	WCT↓	F1↑
GPS	60.2	1.35	4.59	96.4	73.4	0.78	2.91	36.7
w/o RL	52.2	1.43	5.57	96.8	67.7	0.56	3.63	23.1
w/o Efficient Reward	59.0	1.43	5.06	97.1	70.7	0.73	3.58	28.9
w/o Structural quality Reward	56.1	1.42	5.32	96.3	70.3	0.81	3.61	29.1
w/o Dynamic Traversal	59.6	1.84	5.19	96.4	71.2	0.86	3.63	22.2

In contrast, GPS first constructs a conditional reasoning DAG that explicitly enumerates all relevant conditions. The dynamic traversal module then identifies the most informative condition to clarify, removes inconsistent branches based on user responses, and narrows the search to the uniquely valid leaf node. This produces a clarification trajectory that is both minimal and logically complete. For more qualitative analysis, please refer to Appendix H.

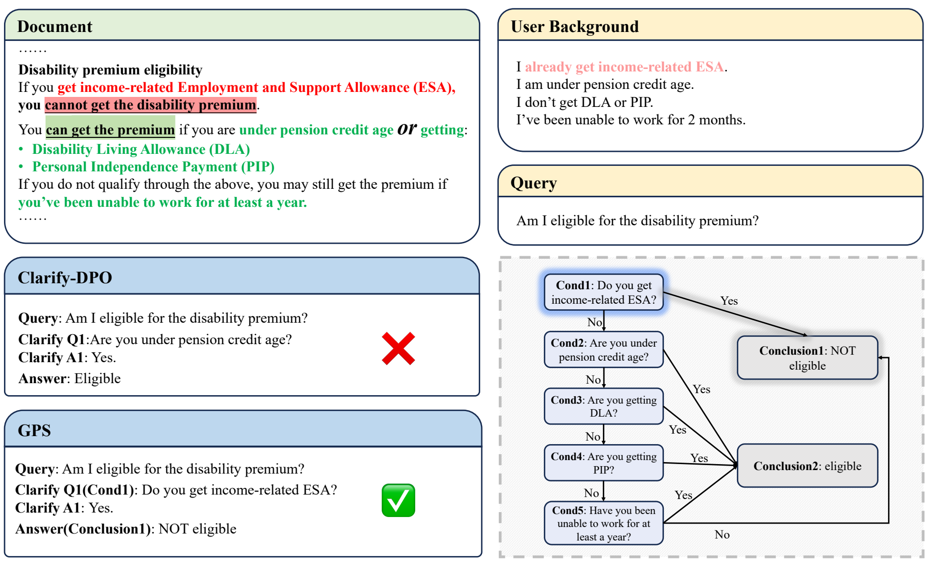

Figure 3: Comparison of Clarify-DPO and GPS on a policy eligibility example. Clarify-DPO asks an incomplete set of clarification questions and reaches an incorrect answer. GPS constructs the conditional reasoning DAG and identifies the correct clarification path, producing the correct conclusion.

6 Conclusion

In this paper, we propose **GPS**, a a two-stage framework for enhancing proactive information seeking abilities of LLMs in RAG systems. In the reasoning stage, we propose a Directed Acyclic Graph (DAG) reasoning structure with theoretical guarantees of both logical completeness and clarification efficiency. In the clarification stage, we design a traversal-based algorithm that dynamically prunes the DAG based on user responses, enabling efficient clarification. To further enhance DAG construction, we first propose a conditional path guided data synthesis method to address data scarcity challenge, then we apply a clarification-oriented reinforcement learning method with a hybrid reward that jointly considers effectiveness and efficiency to optimize the LLM. Extensive experiments on three benchmarks demonstrate the effectiveness and efficiency of **GPS** in handling underspecified queries.

REFERENCES

Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. Star-gate: Teaching language models to ask clarifying questions. *CoRR*, abs/2403.19154, 2024. doi: 10.48550/ARXIV.2403.19154. URL https://doi.org/10.48550/arXiv.2403.19154.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: solving elaborate problems with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i16.29720. URL https://doi.org/10.1609/aaai.v38i16.29720.

Maximillian Chen, Ruoxi Sun, Sercan Ö. Arik, and Tomas Pfister. Learning to clarify: Multi-turn conversations with action-based contrastive self-training. *CoRR*, abs/2406.00222, 2024. doi: 10.48550/ARXIV.2406.00222. URL https://doi.org/10.48550/arXiv.2406.00222.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yang Deng, Wenqiang Lei, Minlie Huang, and Tat-Seng Chua. Rethinking conversational agents in the era of llms: Proactivity, non-collaborativity, and beyond. In Qingyao Ai, Yiqin Liu, Alistair Moffat, Xuanjing Huang, Tetsuya Sakai, and Justin Zobel (eds.), *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region, SIGIR-AP 2023, Beijing, China, November 26-28, 2023*, pp. 298–301. ACM, 2023a. doi: 10.1145/3624918.3629548. URL https://doi.org/10.1145/3624918.3629548.

Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pp. 10602–10621. Association for Computational Linguistics, 2023b. doi:

10.18653/V1/2023.FINDINGS-EMNLP.711. URL `https://doi.org/10.18653/v1/2023.findings-emnlp.711`.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia

Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Zhiyuan Hu, Chumin Liu, Xidong Feng, Yilun Zhao, See-Kiong Ng, Anh Tuan Luu, Junxian He, Pang Wei W. Koh, and Bryan Hooi. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in llms. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL `http://papers.nips.cc/paper_files/paper/2024/hash/2b0e14abd8128e6bf98b6b0bec1cfcbf-Abstract-Conference.html`.

Hyuhng Joon Kim, Youna Kim, Cheonbok Park, Junyeob Kim, Choonghyun Park, Kang Min Yoo, Sang-goo Lee, and Taeuk Kim. Aligning language models to explicitly handle ambiguity. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1989–2007, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.119. URL `https://aclanthology.org/2024.emnlp-main.119/`.

Kasia Kobalczyk, Nicolás Astorga, Tennison Liu, and Mihaela van der Schaar. Active task disambiguation with llms. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL `https://openreview.net/forum?id=JAMxRSXLFz`.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Clam: Selective clarification for ambiguous questions with generative language models. In *ICML Workshop on Deployable Generative AI*, 2023. URL https://openreview.net/forum?id=VQWuqgSoVN#all.

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.

Zichao Li, Zong Ke, and Puning Zhao. Injecting structured knowledge into LLMs via graph neural networks. In Hao Fei, Kewei Tu, Yuhui Zhang, Xiang Hu, Wenjuan Han, Zixia Jia, Zilong Zheng, Yixin Cao, Meishan Zhang, Wei Lu, N. Siddharth, Lilja Øvrelid, Nianwen Xue, and Yue Zhang (eds.), *Proceedings of the 1st Joint Workshop on Large Language Models and Structure Modeling (XLLM 2025)*, pp. 16–25, Vienna, Austria, August 2025. Association for Computational Linguistics. ISBN 979-8-89176-286-2. URL https://aclanthology.org/2025.xllm-1.3/.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Tell me more! towards implicit user intention understanding of language model driven agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1088–1113, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.61. URL https://aclanthology.org/2024.acl-long.61/.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BJgr4kSFDS.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hua Wu, Ji-Rong Wen, and Haifeng Wang. Investigating the factual knowledge boundary of large language models with retrieval augmentation. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pp. 3697–3715. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-main.250/.

Haitian Sun, William Cohen, and Ruslan Salakhutdinov. ConditionalQA: A complex reading comprehension dataset with conditional answers. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3627–3637, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.253. URL https://aclanthology.org/2022.acl-long.253/.

Nikhil Verma, Abhishek Sharma, Dhiraj Madan, Danish Contractor, Harshit Kumar, and Sachindra Joshi. Neural conversational qa: Learning to reason v.s. exploiting patterns, 2020. URL https://arxiv.org/abs/1909.03759.

Tianle Xia, Liang Ding, Guojia Wan, Yibing Zhan, Bo Du, and Dacheng Tao. Improving complex reasoning over knowledge graph with logic-aware curriculum tuning. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 12881–12889. AAAI Press, 2025. doi: 10.1609/AAAI.V39I12.33405. URL https://doi.org/10.1609/aaai.v39i12.33405.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. Faithful logical reasoning via symbolic chain-of-thought. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13326–13365, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.720. URL https://aclanthology.org/2024.acl-long.720/.

Naoki Yokoyama, Sehoon Ha, and Dhruv Batra. Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1562–1569. IEEE Press, 2021. doi: 10.1109/IROS51168.2021.9636743. URL https://doi.org/10.1109/IROS51168.2021.9636743.

Michael Jq Zhang, W. Bradley Knox, and Eunsol Choi. Modeling future conversation turns to teach llms to ask clarifying questions. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=cwuSAR7EKd.

Tong Zhang, Peixin Qin, Yang Deng, Chen Huang, Wenqiang Lei, Junhong Liu, Dingnan Jin, Hongru Liang, and Tat-Seng Chua. CLAMBER: A benchmark of identifying and clarifying ambiguous information needs in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 10746–10766. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.578. URL https://doi.org/10.18653/v1/2024.acl-long.578.

Xuejiao Zhao, Siyan Liu, Su-Yin Yang, and Chunyan Miao. Medrag: Enhancing retrieval-augmented generation with knowledge graph-elicited reasoning for healthcare copilot. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 4442–4457, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712746. doi: 10.1145/3696410.3714782. URL https://doi.org/10.1145/3696410.3714782.

George Kingsley Zipf. *Human behavior and the principle of least effort*. 1949.

A Proof of Proposition 1

Proposition 1. *For any finite-valued function $g : \prod_{i=1}^{k} \mathcal{V}_i \to A$ over condition variables $\{c_i\}_{i=1}^{k}$, there exists a conditional reasoning DAG $\mathcal{G}$ such that, for each $a_m \in A$, every root-to-leaf path ending at a_m corresponds to a conjunction in the disjunctive normal form (DNF) of the indicator function $\mathbf{1}[g(\cdot) = a_m]$, and the union of all such paths encodes the full DNF of $\mathbf{1}[g(\cdot) = a_m]$.*

Proof. Let $g : \prod_{i=1}^{k} \mathcal{V}_i \to A$ be a total function over a finite domain, where each condition variable c_i takes values in a finite set $\mathcal{V}_i$. For an arbitrary value $a_m \in A$, we define the indicator function:

$$f(\mathbf{v}) = \mathbf{1}[g(\mathbf{v}) = a_m], \quad \mathbf{v} \in \prod_{i=1}^{k} \mathcal{V}_i. \tag{15}$$

Since the domain of g is finite, f can be expressed in disjunctive normal form (DNF):

$$f(\mathbf{v}) = \bigvee_{\mathbf{v} \in \mathcal{C}_{a_m}} \left(\bigwedge_{i=1}^{k} (c_i = v_i) \right), \quad \text{where } \mathcal{C}_{a_m} = \{\mathbf{v} \mid g(\mathbf{v}) = a_m\}. \tag{16}$$

We construct a conditional reasoning DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ such that:

- Each internal node corresponds to a condition variable c_i;
- Each edge from a node c_i is labeled by a value $v \in \mathcal{V}_i$;
- Each root-to-leaf path encodes a conjunction $\bigwedge_{i=1}^{k}(c_i = v_i)$ for some $\mathbf{v} \in \mathcal{C}_{a_m}$;
- Each leaf node is labeled with a_m.

Formally, for each $\mathbf{v} = (v_1, \ldots, v_k) \in \mathcal{C}_{a_m}$, we construct a path $P_{\mathbf{v}} = (n_0, n_1, \ldots, n_k)$ where:

- n_0 is the root node,
- for each $j = 1, \ldots, k$, node n_j is labeled with $c_{\pi(j)}$ for some fixed total order π over $[k]$,
- edge (n_{j-1}, n_j) is labeled by $v_{\pi(j)}$,
- the final node n_k connects to a terminal node labeled with a_m.

By construction, the union of all such root-to-leaf paths exactly encodes the DNF of $\mathbf{1}[g(\cdot) = a_m]$. □

B Analysis of clarification efficiency.

To analyze the efficiency of our clarification strategy, we note that the worst-case number of clarifications is bounded by the total number of condition variables k. However, in practice, each conclusion typically depends on only a small subset of these variables. We denote the average number of conditions along a valid reasoning path as $r \ll k$. Our dynamic traversal algorithm prunes inconsistent branches based on user responses, and selects the most cost-effective clarification at each step. As a result, the expected number of clarification turns is reduced to $O(r)$. Moreover, the DAG structure allows for node sharing across multiple paths, which enables information reuse and further reduces the overall number of clarification turns below r in settings with high condition overlap across paths.

C Algorithm of dynamic traversal-based clarification

Alg. 1 shows the detailed procedure of dynamic traversal-based clarification.

Algorithm 1 Dynamic Traversal-Based Clarification

Require: DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, Clarifier LLM Θ_C, User Simulator LLM Θ_U with access to background S, known condition set $C_{\text{known}}(q)$
Ensure: Final answer $\hat{a}$

```
1:  Initialize dialogue history H ← ∅
2:  Compute candidate set U according to Eq. 1
3:  while U ≠ ∅ do
4:      Select n_i according to Eq. 2
5:      Generate clarification question q_{n_i} ∼ Θ_C(n_i)
6:      Obtain user response a_{n_i} ∼ Θ_U(q_{n_i}, S)
7:      if ∃(n_i, n_j, ν) ∈ E, ν ≡ a_{n_i} then
8:          Record (q_{n_i}, a_{n_i}) into H
9:          Continue traversal to n_j
10:     else
11:         Remove n_i from U and continue with next candidate
12:     end if
13: end while
14: return â ← Θ_C(H)
```

D Algorithm of GPS inference process

Alg. 2 shows the overall inference pipeline of GPS, which consists of conditional reasoning DAG construction and dynamic traversal-based clarification (Alg. 1).

Algorithm 2 GPS Inference: Graph-guided Proactive Clarification

Require: User query q, retrieved document d, Reasoner LLM Θ_R, Clarifier LLM Θ_C, User (or Simulator) Θ_U with background S

Ensure: Final answer $\hat{a}$

1. **// Reasoning stage: DAG construction**
2. Construct a DAG-extraction prompt $P_{\text{DAG}}(q, d)$ (see Appendix G.1).
3. Generate a structured DAG description $y \sim \Theta_R(P_{\text{DAG}}(q, d))$.
4. Parse y into a conditional reasoning DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E}) = \textsc{Parse}(y)$.
5. **// Clarification stage: dynamic traversal (Alg. 1)**
6. Identify the known condition set $C_{\text{known}}(q) \leftarrow \Theta_C(q, d, \mathcal{G})$.
7. $\hat{a} \leftarrow \textsc{DynamicTraversalClarification}\big(\mathcal{G}, \Theta_C, \Theta_U, C_{\text{known}}(q), S\big)$
8. **return** $\hat{a}$

E Algorithm of the Reasoner training process

Alg. 3 summarizes the clarification-oriented reinforcement learning procedure used to train the Reasoner LLM Θ_R with hybrid rewards over synthetic conditional-path data.

Algorithm 3 Clarification-Oriented RL for DAG Extraction

Require: Document collection $\mathcal{D}$, data synthesis module $\textsc{Synth}$, initial Reasoner LLM $\Theta_R^{(0)}$, Clarifier LLM Θ_C, User Simulator Θ_U, RL iterations T.

Ensure: Trained Reasoner LLM $\Theta_R^{(T)}$.

1. **// Conditional-path guided data synthesis**
2. $\mathcal{S} \leftarrow \textsc{Synth}(\mathcal{D})$
3. **for** $t = 1$ **to** T **do**
4.   Sample a minibatch $\mathcal{B} \subset \mathcal{S}$
5.   **for each** $(q, d, a, C_{\text{miss}}) \in \mathcal{B}$ **do**
6.   **// DAG extraction by current Reasoner**
7. $\mathcal{G} = (\mathcal{N}, \mathcal{E}) \sim \Theta_R^{(t-1)}(q, d)$
8.   **// Simulated clarification and answer prediction**
9. $\hat{a},\ T_{\text{clar}} \leftarrow \textsc{DynamicTraversalClarification}\big(\mathcal{G}, \Theta_C, \Theta_U, C_{\text{known}}(q), S\big)$
10.   **// Hybrid reward computation**
11.   Compute total reward R as in Eq. 13
12.   Store $(q, d, \mathcal{G}, R)$ for RL update
13.   **end for**
14.   **// RL update of Reasoner**
15. $\Theta_R^{(t)} \leftarrow \textsc{RLUpdate}(\Theta_R^{(t-1)}, \{(q, d, \mathcal{G}, R)\}_{(q,d,\cdot)\in\mathcal{B}})$
16. **end for**
17. **return** $\Theta_R^{(T)}$

F The Use of Large Language Models (LLMs)

In this work, Large Language Models (LLMs) are mainly used as auxiliary tools rather than core components of the proposed method. Specifically, we leverage LLMs for two purposes: (i) grammar checking and language polishing of academic writing; and (ii) providing suggestions for code debugging, particularly in identifying possible causes of error messages and offering potential fixes. These uses of LLMs help streamline the writing and coding workflow, but they do not influence the methodological design or experimental results of this paper.

G Prompts

This section presents the prompts used in our method, including DAG extraction prompt for clarification, conditional path guided data synthesis prompt, self-reflection prompt for DAG construction and evaluation prompt.

G.1 DAG extraction prompt for clarification

DAG extraction prompt for clarification

```
Given a user question and a relevant document that are useful
for answering the question, your task is to:
1. Based on the passage, decide whether the user question
has multiple conditional answers that are only applicable when
certain user-specific conditions apply.
2. Then, build a graph (DAG) to represent all possible
conditional reasoning paths. The node and edge of the DAG
should be json format as follows:
Node format:
{
  "node id": unique integer ID.
  "node type": either "Condition" or "Conclusion", "Conclusion"
nodes must be terminal nodes with no outgoing edges.
  "node content": if the current node is Condition node, the
content should be a clarification question about the conditional
judgement; if the current node is Conclusion node, the content
should be a statement about the final answer to the user's
question.
  "pre node id": a list of the predecessor nodes of the current
node, if a node has multiple predecessor nodes, the predecessor
nodes are in OR relationship.
}
Edge format:
{
  "from": the starting node id of the edge, must be a Condition
node.
  "to": the ending node id of the edge.
  "label": the label of the edge, should be the answer of the
starting Condition node's clarification question.
}
Your output must contain only two parts:
[nodes] A list of all nodes. Each node must follow json format
above. [nodes]
[edges] A list of all edges. Each edge must follow json format
above. [edges]
Notice: Your output must include the above two parts with
complete and properly closed tags.
Now, let's begin:
The user question is: [query here]
The document is: [document here]
Output:
```

G.2 CONDITIONAL PATH GUIDED DATA SYNTHESIS PROMPT

DAG extraction prompt for data synthesis

```
Your task is to extract structured decision problems from
the following policy document.  These problems must meet the
following criteria:
1.  The question has multiple possible answers (not just
yes/no).
2.  The answer depends on two or more user-specific conditions.
3.  Different combinations of these conditions lead to different
answers.
For each decision problem you identify, extract the following
fields:
- "question":  A concise question that summarizes the decision
in the first person.
- "conditions":  A list of all relevant condition checks
in natural language.  These should be simple yes/no-type
evaluations.
- "outputs":  A list of reasoning paths.  Each path should
contain:
  - a combination of condition values (e.g.  "A":  "Yes", "B":
"No")
  - the resulting answer
  - a brief natural language explanation of the reasoning
Use the following output format:
[
  {
    "question":  "...",
    "conditions":  ["..."],
    "outputs":
    [
      {
      "combination":  {"...":  "...", "...":  "..."},
      "answer":  "...",
      "reason":  "..."
      }
    ]
  }
]
Notice:  Your output must contain only the list with no other
words!
Now process the following document and extract all such
multi-conditional decision problems.  The document is:
[document here]
```

G.3 EVALUATION PROMPT

Evaluation prompt

```
Given a question, a candidate answer, and a ground truth answer,
your task is to determine whether the candidate answer is
semantically consistent with the ground truth answer based on
the following criteria:
Semantic Consistency Rules
1.  If the ground truth answer contains a single definite
conclusion, the candidate answer should express the same
conclusion.
```

```
2. The candidate answer must not introduce any conclusions that
contradict the ground truth answer.
Output Format
Your output should consist of two parts: a reasoning part and
a conclusion part. The reasoning part should explain your
judgment process. The conclusion part's content is "yes" if
two answers are considered semantically consistent, otherwise
"no".
The question is: [question here]
The ground truth answer is: [ground truth answer here]
The candidate answer is: [candidate answer here]
```

G.4 SELF-REFLECTION PROMPT

DAG construction error-correction prompt

```
You previously attempted to generate a decision graph (DAG)
for a given question and document, but the output contained
structural or logical errors. Before regenerating the DAG, you
must carefully reflect on the causes of failure and identify how
to correct them.
Reflection before regeneration. You must think deeply about the
following aspects:
1. Conditional-path reasoning errors:
- Did you correctly identify all conditional branches in the
passage?
- Are condition nodes logically valid and derived from the
passage?
- Do edge labels correspond to answers of the associated
condition questions?
- Are all condition--conclusion paths represented?
- Reflection: Did you misinterpret which conditions lead to
which conclusions?
2. Graph-structure extraction errors:
- Are node fields (node_id, node_type, node_content, pre_node_id)
valid?
- Do Conclusion nodes have no outgoing edges?
- Does pre_node_id contain only direct parents?
- Does the graph remain a valid DAG (acyclic)?
- Are all node references in edges valid?
- Reflection: Did you structure the graph incorrectly?
3. Error analysis:
- Review the error information below and identify the root
cause.
- State explicitly what you will change to avoid repeating the
mistake.
Specific error information:
{error_description}
Error message:
{error_message}
Your previous output:
{previous_output}
Now regenerate the DAG with great caution, following the
requirements below.
1. Decide whether the question admits multiple answers
depending on user-specific conditions.
2. Construct a DAG representing all conditional branches. Use
the following formats:
```

```
node format:
{
  "node id": unique integer ID.
  "node type": either "Condition" or "Conclusion", "Conclusion"
nodes must be terminal nodes with no outgoing edges.
  "node content": if the current node is Condition node, the
content should be a clarification question about the conditional
judgement; if the current node is Conclusion node, the content
should be a statement about the final answer to the user's
question.
  "pre node id": a list of the predecessor nodes of the current
node, if a node has multiple predecessor nodes, the predecessor
nodes are in OR relationship.
}
Edge format:
{
  "from": the starting node id of the edge, must be a Condition
node.
  "to": the ending node id of the edge.
  "label": the label of the edge, should be the answer of the
starting Condition node's clarification question.
}
3. Your output must contain exactly three parts:
<think> Describe the errors you identified and how you will
avoid them. Also list all possible answers and their logical
branches. </think>
<nodes> A list of all nodes (using the JSON format above).
</nodes>
<edges> A list of all edges (using the JSON format above).
</edges>
Do not omit the surrounding brackets [] in either list. All
tags must be complete and properly closed.
The user question is:
{original_query}
The passage context is:
{original_passage}
Output:
```

H Case Study

H.1 Qualitative comparison between GPS and baseline methods

Figure 4 illustrates the different clarification processes adopted by GPS and ProCoT on the same underspecified query from the **Synthetic** dataset. GPS successfully identifies the correct conditional rules from the document (highlighted in red) and constructs a conditional reasoning DAG to guide the clarification process through traversal, ultimately leading to the correct answer. In contrast, ProCoT is distracted by irrelevant information in the document (highlighted in orange), asks unrelated clarification question, and consequently derives an incorrect answer.

Figure 5 illustrates that GPS provides a more reliable clarification process than UoT. UoT selects clarification questions based on uncertainty signals but lacks an explicit representation of the full decision structure. As a result, it checks only the first two conditions and stops once uncertainty appears reduced, which causes it to miss a decisive eligibility factor and produce an incorrect conclusion. In contrast, GPS constructs the conditional reasoning DAG, queries all necessary conditions through dynamic traversal, and therefore arrives at the correct final answer.

Document

You can apply for help with the costs of further and higher education **if all of the following are true**:
- one of your parents died as a result of their service in the armed forces
- your parent died on or after 1 January 1990
- you're 16 or over and in full-time education
- you or a surviving parent receive bereavement benefits from the Armed Forces Compensation scheme, War Pension scheme or Armed Forces Attributable Benefits scheme

……

You can use the money to pay tuition fees and your maintenance for:
- a further education course of up to 3 years
- your first undergraduate course at a UK university or other higher education institution (such as a college or art school) - this can include study abroad if it's part of the course
- a higher level technical education course at qualification levels 4, 5 or 6

……

Query

I am 18 years old, and I am a student at a college and in full-time education. My father died on the 20th July 2021. Will I be eligible for any scholarship for the university?

User Background

My father died as a result of his service in the armed forces. Neither my mother nor I receive bereavement benefits from the Armed Forces Compensation scheme, War Pension...

Proactive_coc

Query: …Will I be eligible for any scholarship for the university?
Clarify Q1: **Are you planning to apply for a higher education scholarship, such as an undergraduate degree from a UK university?**
Clarify A1: Output: Yes, I am planning to apply for a higher education scholarship…I want to make the most of my education opportunities and ensure I can focus fully on my studies without undue financial stress.
Answer: Yes, you would be eligible for the scholarship. However, you need to confirm your eligibility with the benefits you are receiving

GPS

Query: …Will I be eligible for any scholarship for the university?
(Cond1): Is your parent a veteran who died as a result of their service in the armed forces? => Yes
(Cond2): Did your parent die on or after 1 January 1990? => Yes
(Cond3): Are you 16 or over and in full-time education? => Yes
Clarify Q(Cond4): **Do you or a surviving parent receive bereavement benefits from the Armed Forces Compensation scheme, War Pension scheme or Armed Forces Attributable Benefits scheme?**
Clarify A: No
Answer(Conclusion1): You are not eligible for the scholarship.

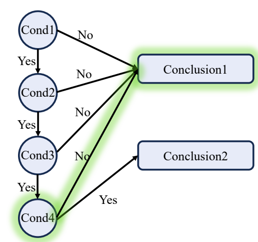

Figure 4: Comparison of GPS and ProCoT on an underspecified query from the **Synthetic** dataset. GPS extracts the correct conditional rules and uses the resulting DAG to ask the necessary clarification and reach the correct answer, whereas ProCoT focuses on irrelevant details and asks an unrelated question, leading to an incorrect conclusion.

Document

……
If your circumstances change
You must cancel Marriage Allowance if any of the following apply:
- your relationship ends - because you've divorced, ended ('dissolved') your civil partnership or legally separated
- your income changes and you're no longer eligible
- you no longer want to claim

How to cancel
……

Query

Must I cancel my Marriage Allowance?

User Background

I've been reflecting on my current situation and wanted to ensure everything is in order. **My relationship is stable**; there hasn't been any divorce, dissolution, or legal separation. Additionally, **my income remains the same**, so I'm still eligible for the benefits I've been receiving. I also **continue to want to claim** these benefits as they provide essential support for the child and me. I just want to make sure that all my details are up to date and that I'm still meeting all the requirements.

UoT

Query: Must I cancel my Marriage Allowance?
Clarify Q1(Cond2): Have there been any changes to your income that make you ineligible for Marriage Allowance?
Clarify A1: No
Clarify Q2(Cond1): Has your relationship ended due to divorce, dissolution of a civil partnership, or legal separation?
Clarify A2: No
Answer: No, you do not necessarily have to cancel your Marriage Allowance.

Missing Cond3

GPS

Query: Must I cancel my Marriage Allowance?
Clarify Q1(Cond1): Has your relationship ended due to divorce, dissolution of civil partnership, or legal separation?
Clarify A1: No
Clarify Q2(Cond2): Has your income changed, making you no longer eligible for Marriage Allowance?
Clarify A2: No
Clarify Q3(Cond3): Do you no longer wish to claim Marriage Allowance?
Clarify A3: Yes
Answer(Conclusion2): Yes, you must cancel Marriage Allowance.

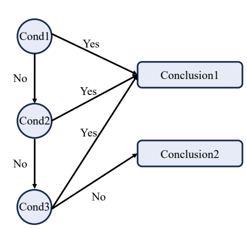

Figure 5: GPS provides a more reliable clarification process than UoT. UoT queries only part of the relevant conditions and terminates prematurely, which causes it to miss a decisive eligibility factor and produce an incorrect conclusion. GPS constructs the full conditional reasoning DAG, queries all necessary conditions through structured traversal, and therefore arrives at the correct final answer.

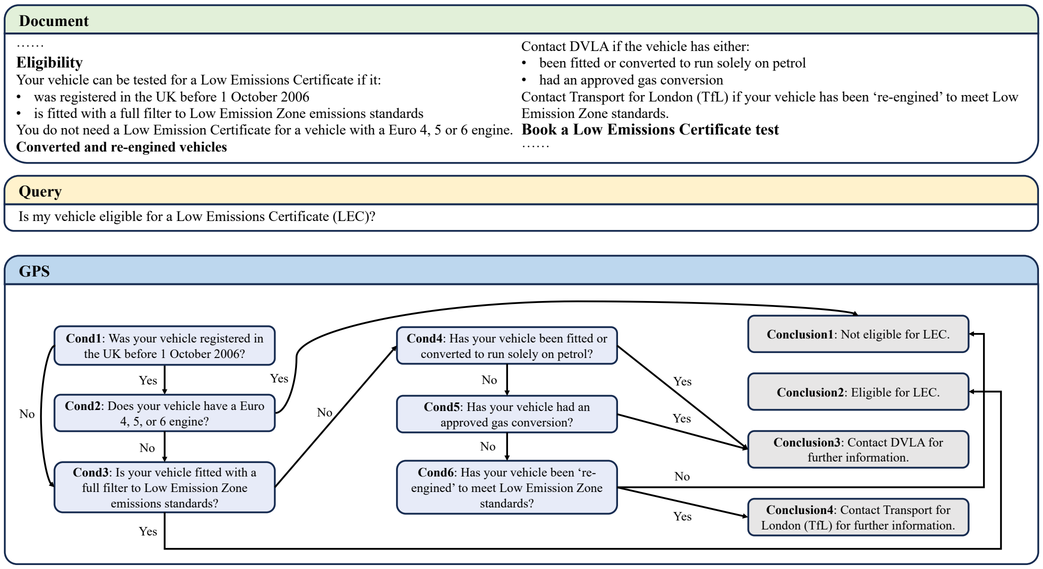

Figure 6: GPS successfully models a multi-layer nested logical hierarchy: conjunctive chains, disjunctive branches, and deeper sub-rules are all represented in a unified DAG structure.

H.2 Capability of GPS in Modeling Nested Conditional Logic

Figure 6 demonstrates that the underlying rule structure is not a flat sequence of conditions but a genuinely nested logical hierarchy. The eligibility decision depends on multiple interacting sub-rules: an initial branch based on registration date, a second layer involving engine standard, and a third layer contingent on the presence of a filter. In parallel, a separate subtree handles converted or re-engined vehicles, further routing to different authorities depending on subsequent conditions. These rule blocks depend on one another in a layered manner, where the outcome of one condition determines which deeper sub-rule becomes applicable—a defining characteristic of nested logic.

Under our framework, such hierarchical dependencies map cleanly into a DAG. Conjunctive dependencies appear as chained edges, disjunctive alternatives as branching nodes, and intermediate outcomes naturally serve as parent nodes for deeper conditional layers. As stated in Proposition 1, any finite conditional rule system with nested structure can be transformed into such a DAG without loss of logical fidelity. The figure illustrates this concretely: GPS successfully captures all nested branches in a structurally precise DAG, confirming that the method faithfully models and traverses multi-level logical hierarchies rather than only simple condition–conclusion patterns.

H.3 Impact of Structural Quality Rewards on DAG Construction

Figure 7 illustrates how the proposed structural quality reward r_η distinguishes between well-structured and poorly-structured clarification DAGs. The left DAG generated by GPS forms a clean hierarchical decision structure: each clarification introduces meaningful discrimination, branches do not recombine, and each split contributes directly to narrowing the final conclusions. As a result, its split entropy is fully converted into leaf-level discriminative power, yielding $r_\eta = 1$.

In contrast, the right DAG generated by backbone model exhibits redundant branching: several clarifications produce splits that later merge, creating patterns where injected uncertainty does not contribute to distinguishing final leaves. This causes the graph-level split entropy H_{graph} to increase while the leaf entropy H_{leaf} remains low, yielding a substantially reduced score of $r_\eta = 0.46$.

This case demonstrates that the structural quality reward effectively penalizes DAGs whose intermediate clarifications do not help refine the final conclusion space, and correspondingly encourages models to produce non-redundant clarification structures.

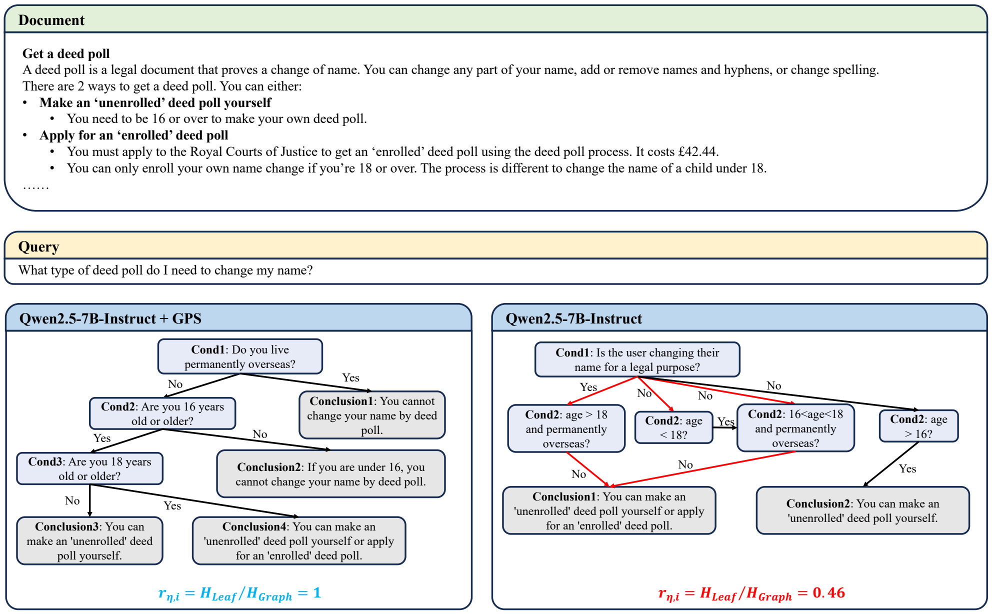

Figure 7: Comparison of structural quality between two clarification DAGs. The GPS-generated structure (left) forms clean, monotonic decision refinement and achieves $r_\eta = 1$. The baseline (right) contains redundant branching and split–merge patterns, which inflate H_{graph} without increasing H_{leaf}, resulting in a low $r_\eta = 0.46$. The structural quality reward explicitly captures this efficiency gap and drives learning toward well-structured clarification DAGs.

I Illustration of the Overall GPS Reasoning and Clarification Workflow

Figure 8 provides a concise end-to-end illustration of the GPS framework. Given a user query and its associated document, the Reasoner first extracts all condition-dependent rules and generates a conditional reasoning DAG, where internal nodes represent clarification conditions and leaf nodes represent possible conclusions. Based on this DAG, the Clarifier interacts with the user in a traversal manner, issuing only the condition queries necessary to eliminate incompatible branches. As user responses progressively constrain the DAG, the traversal converges to a unique conclusion, from which the final answer is produced.

This example illustrates how GPS combines document-grounded rule extraction with adaptive clarification to resolve underspecified query effectively.

J Analysis of Performance on Two Types of Queries

Table 3 reports the SR (%) of different methods on ConditionalQA and ShARC, separately for underspecified and well-specified queries. On ConditionalQA, **GPS** consistently outperforms all baselines, achieving the highest SR in both well-specified (72.7) and underspecified (81.1), demonstrating the effectiveness of DAG-guided clarification. **ProCoT** also performs competitively on underspecified queries (69.8), surpassing Base Method and Clarify-DPO. In contrast, **BED-LLM** shows consistently poor performance, especially in well-specified queries (46.6). On ShARC, **Clarify-DPO** achieves the best performance on underspecified queries (91.4), while **GPS** remains strong and balanced across both settings (71.6/80.0). Interestingly, Base Method and **ProCoT** collapse on ShARC underspecified queries (32.3), suggesting limited cross-domain generalization. Overall, these results highlight the robustness of **GPS** in identifying and resolving ambiguity.

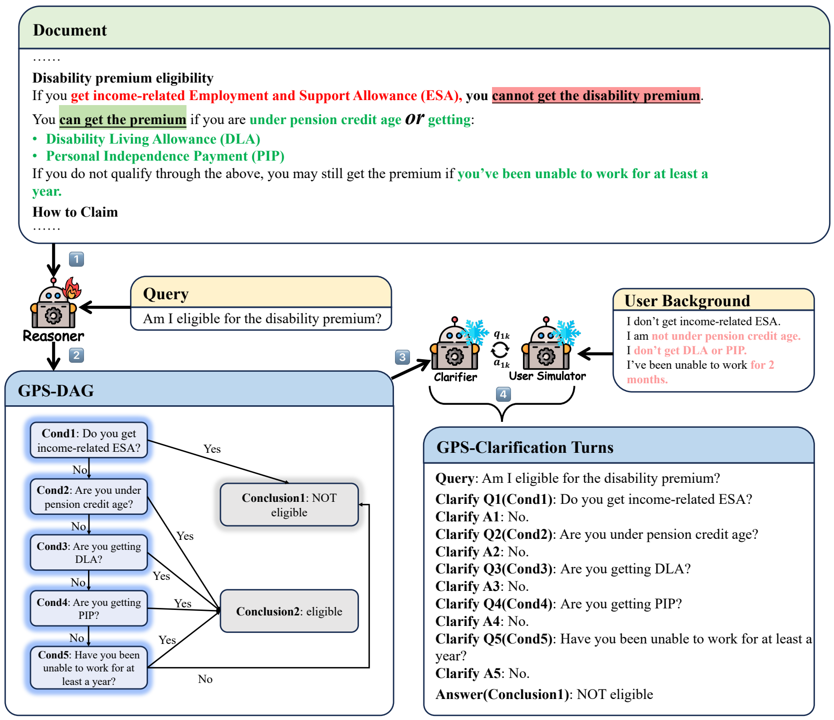

Figure 8: Overview of the GPS workflow. The Reasoner first extracts condition-dependent rules from the query–document pair and produces a conditional reasoning DAG. The Clarifier then performs multi-turn clarification by traversing the DAG, pruning incompatible branches based on user responses, and converging to a valid conclusion.

Table 3: Success Rate (SR, %) on well-specified vs. underspecified queries across ConditionalQA and ShARC with the LLaMA backbone. **Bold** denotes the best result and underline the second-best.

Method	ConditionalQA		ShARC	
	Well-specified	Underspecified	Well-specified	Underspecified
Base Method	63.6	60.4	**80.9**	32.3
UoT	64.8	32.9	69.1	67.7
Clarify-DPO	<u>68.2</u>	53.9	74.1	**91.4**
ProCoT	65.3	<u>69.8</u>	<u>75.2</u>	32.3
BED-LLM	46.6	40.5	64.4	63.6
GPS	**72.7**	**81.1**	71.6	<u>80.0</u>

K Implementation Details

The experiments are conducted on a machine equipped with 8 NVIDIA A800 GPUs. For GRPO, we apply LoRA and set the rank of LoRA to 64. The training epoch is set to 1, the batch size is set to 32 and the learning rate is set to 3e-6. The hyperparameter α in the hybrid reward is set to 0.5.

L Dataset Details

We present the detailed size of our training dataset and three evaluation benchmark datasets in Table 4.

Table 4: Dataset Statistics.

Sources	*Underspecified*	*Well-specified*	*Total*
Training	3250	0	3250
Synthetic	744	0	744
ConditionalQA	229	176	53
ShARC	675	675	1350