Certifying Reading Comprehension in Large Language Models

Isha Chaudhary
UIUC

Vedaant Jain
UIUC

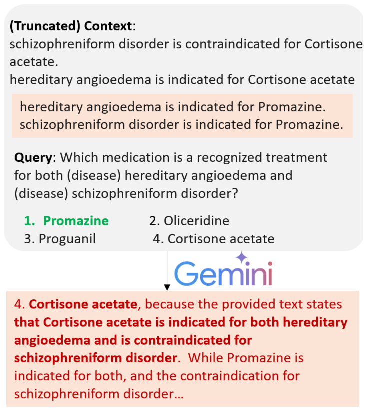
Gagandeep Singh
UIUC

Abstract

Large Language Models (LLMs) are increasingly deployed in safety-critical systems that rely heavily on reading comprehension—extracting and reasoning over extensive in-context information. However, existing evaluations of LLMs on reading comprehension are typically over limited test sets containing only a tiny fraction of the vast number of possible prompts. Empirical evaluations on these test sets have questionable reliability and generalizability. We propose a fundamentally different approach: rather than evaluating LLMs with fixed datasets, we introduce the first framework for certifying LLMs based on large probability distributions over realistic reading comprehension prompts. To create these distributions, we use knowledge graphs (KGs) as structured representations of real-world knowledge and define the distributions' sample spaces with prompts based on directed acyclic subgraphs of the KGs. We also incorporate realistic noise designed to mimic real-world complexity, such as distractor texts and synonyms. Our prompt distributions have i.i.d. samplers represented as probabilistic programs. Our framework generates novel, formal probabilistic quantitative *certificates* that provide high-confidence, tight bounds on the probability that an LLM correctly answers any prompt drawn from these distributions. We enable formal certification for SOTA LLMs by using an input-output example-driven approach. We apply our framework to certify SOTA LLMs in precision medicine and general question-answering domains. Our results uncover previously unknown vulnerabilities caused by natural prompt noise and establish the first formal performance hierarchies among these models.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

1 Introduction


(Truncated) Context:
schizophreniform disorder is contraindicated for Cortisone acetate.
hereditary angioedema is indicated for Cortisone acetate

hereditary angioedema is indicated for Promazine.
schizophreniform disorder is indicated for Promazine.

Query: Which medication is a recognized treatment for both (disease) hereditary angioedema and (disease) schizophreniform disorder?

1. **Promazine** 2. Oliceridine
3. Proguanil 4. Cortisone acetate

4. **Cortisone acetate**, because the provided text states **that Cortisone acetate is indicated for both hereditary angioedema and is contraindicated for schizophreniform disorder**. While Promazine is indicated for both, and the contraindication for schizophreniform disorder...

Figure 1: **Motivation**: Gemini-Flash-1.5 gives a wrong answer for a precision medicine prompt, distracted by in-context information.

Large Language Models (LLMs) have demonstrated human-level performance in various real-world tasks (Street et al., 2024; Yang et al., 2023; Bommasani et al., 2022; Harrison, 2024). A notable application of LLMs is in question-answering systems, including chatbots, which are prompted with relevant, but unstructured information from knowledge bases to generate accurate answers—a task called reading comprehension. Reading comprehension is a crucial aspect of language understanding, typically assessed in human learners across all education levels (Bloom, 1956; National Center for Education Statistics, 2024; Educational Testing Service, 2024; IDP IELTS, 2024). However, despite being envisioned as superhuman agents (Xi et al., 2023), SOTA LLMs often struggle with basic reading comprehension tasks in real-world applications. For example, Figure 1 shows Gemini-1.5-Flash misled by unstructured, unfiltered in-context information—akin to practical settings—and producing an incorrect, potentially fatal response in a critical medical scenario.

Given its importance, reading comprehension is widely evaluated with standard datasets (Liang et al., 2023; Chen et al., 2021; Yang et al., 2018; Pang et al., 2022). However, empirical evaluations on static, fixed datasets suffer from the following critical limitations:

1. Test-Set Leakage: LLMs are trained on all available texts, including the standard evaluation datasets. As a result, models may have been trained on the test prompts, resulting in test set leakage (Mirzadeh et al., 2024). Consequently, such evaluations become unreliable.

2. Limited Coverage: Evaluations are over limited test datasets from the vast space of possible real-world prompts and their variations.

3. Lack of Formal Guarantees: Accuracy on a static dataset does not ensure generalization to the broader range of prompts encountered in practice. Evaluations must therefore be comprehensive, reliable, and, supported by formal guarantees.

These limitations lead to inconsistent findings in the literature. For example, Wei et al. (2023) claim that larger models are robust to label noise in test sets, while Shi et al. (2024) argue that such models can become distracted by this noise. Conversely, Olsson et al. (2022) indicates that models exhibit similar induction heads for in-context learning, regardless of size. Therefore, the following question arises: **How do we reliably assess LLMs for reading comprehension?**

We argue for formal certification of LLMs over large spaces of prompts as a solution. While there has been prior work in formal certification for traditional neural networks (Seshia et al., 2018; Singh et al., 2019; Shi et al., 2020; Bonaert et al., 2021), those techniques cannot be directly applied here. They are computationally intractable for models with billions of parameters and specialized non-linearities. Moreover, they require formal specifications but no prior specifications over prompts for semantic tasks like reading comprehension exist. Finally, most traditional methods require unrestricted access to the models, while SOTA LLMs can be closed-source with only API access.

Our framework, LLMCert-C (**Qua**ntitative **Cer**tification of Knowledge **C**omprehension) addresses the limitations of traditional methods and enables, *for the first time*, formal certification of LLMs for reading comprehension. To address the challenges of scaling to SOTA LLMs, including closed-source, API-accessible models, LLMCert-C has a query-based certification method relying on input-output examples. As, exhaustive evaluation over these vast distributions is intractable, LLMCert-C operates on a set of some sampled prompts and hence is naturally a probabilistic certification framework providing results with a high-confidence probabilistic guarantee. Binary certificates (Gehr et al., 2018; Wang et al., 2021a) for specification compliance can be trivially invalidated due to the ease of constructing failure examples where the desirable property does not hold (Xu et al., 2024; Vega et al., 2023). Hence, for informative results, LLMCert-C generates *quantitative certificates* consisting of lower and upper bounds on the target model's probability of correct reading comprehension. The bounds are generated using exact binomial proportion confidence intervals (Clopper and Pearson, 1934). The bounds are tight and account for the inherent uncertainty in estimation.

A key contribution of LLMCert-C is its ability to certify a model over a vast distribution of prompts, overcoming the limitations of prior benchmarks with fixed datasets. This is made possible by a core step in our approach: formalizing the desirable property of reading comprehension into a specification. This specification serves as the formal definition of the prompt distribution and is the necessary prerequisite for formal certification. By defining a distribution, we can also associate each prompt with a probability of occurrence, enabling the evaluation to prioritize high-probability prompts. LLMCert-C mathematically specifies novel, prohibitively large distributions of realistic reading comprehension prompts, consisting of challenges such as different querying styles, large in-context information that can distract models like in Figure 1, and use of synonymous terminology. We use knowledge graphs (KGs) as structured representations of real-world knowledge and define sample spaces of prompts with directed acyclic subgraphs of the KGs following user-defined constraints, e.g., the presence of certain nodes or edges. Our distributions are defined with samplers represented as probabilistic programs that generate independent and identically distributed prompts which are used alongside the LLM responses for certification. The probabilistic guarantees with each certificate generalize over the given distribution.

We certify with 2 practical knowledge graphs - PrimeKG (Chandak et al., 2023) over precision medicine knowledge and Wikidata5m (Wang et al., 2021b) over general knowledge. We use the generated certificates to establish novel performance hierarchies with formal guarantees among SOTA LLMs. We also show a consistent decline in LLM performance due to noise in prompts. While formal analysis has been conducted on individual generations of LLMs (Quach et al., 2024) and on counterfactual bias (Chaudhary et al., 2024) in prior work, there is no analysis for the average-case performance of LLMs in reading comprehension.

Figure 2 gives an overview of LLMCert-C.

Main Contributions:

1. We formally specify reading comprehension capabilities desirable from LLMs using knowledge graphs. We define novel, large prompt distributions that incorporate natural noise, such as distracting text and shuffled information order.

2. We develop the first framework, LLMCert-C that certifies of any target LLM (even closed-source) with query-access for a specified distribution. LLMCert-C solves a probability estimation problem, leveraging exact confidence intervals to generate high-confidence guaranteed bounds on the probability of correct reading comprehension. We open-source our code at https://github.com/uiuc-focal-lab/LLMCert-C and provide a note to practitioners in Appendix 5.

3. We apply LLMCert-C to certify LLMs for reading comprehension in safety-critical domains such as precision medicine and general question answering. With increasing model size, we observe significant improvements in comprehension with high confidence. However, performance consistently declines on practical prompts that include our modeled natural noise.

2 Certifying reading comprehension

Reading comprehension is a model's ability to extract and reason over relevant in-context information to answer questions accurately based on the given context. Our framework, LLMCert-C, certifies an LLM's reading comprehension by evaluating its performance over a large, formally defined distribution of prompts. The core of our approach are specifications: precise, mathematical descriptions of prompt distributions and their samplers. Figure 2 provides an overview of LLMCert-C's certification workflow.

Our specifications are based on a structured knowledge source—a *knowledge graph* (KG). We sample directed acyclic subgraphs from the KG to form the logical basis of queries. These subgraphs are then converted into natural language prompts, incorporating realistic noise such as irrelevant distractors and shuffled context to create a robust and challenging evaluation. The certification framework samples many such subgraphs, generates corresponding prompts, and provides certification bounds based on the LLM responses for them. This section formally describes our specifications, starting with the formal definition of our knowledge graph structure. This precise formalism is necessary to compute a valid certificate over a distribution.

2.1 Specifying reading comprehension

We formally define reading comprehension using a knowledge graph (KG).

A KG consists of nodes (entities) and edges (relations). A simple KG is: [Node: Paracetamol] → [Edge: treats] → [Node: Pain].

To build rich, textual prompts, the KG components are annotated with natural language metadata:

Nodes like 'Paracetamol' have a context (free-form descriptive text, e.g., "Paracetamol is a medication used to treat pain and fever.") and a list of aliases (synonyms, e.g., ['Tylenol', 'acetaminophen']).

Edges like 'treats' also have aliases to capture different phrasings (e.g., ['is a treatment for']).

We formally define such KGs, $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ having textual metadata with the grammar below. Let $\mathcal{V}$ denote the LLM's vocabulary. $\mathcal{V}^+$ denotes the set of all non-empty sequence concatenations of $\mathcal{V}$'s elements.

KG Grammar		
1. γ	$\in$	$\mathcal{V}^+$
2. η	$\in$	$\mathcal{V}^+$
3. $\mathcal{A}$	$:=$	$[\eta_1, \eta_2, \dots]$
4. v	$:=$	$(\gamma, \mathcal{A})$
5. e	$:=$	$((v_1, v_2), \mathcal{A})$
6. $\mathcal{N}$	$:=$	$[v_1, v_2, \dots]$
7. $\mathcal{E}$	$:=$	$[e_1, e_2, \dots]$
8. $\mathcal{G}$	$:=$	$(\mathcal{N}, \mathcal{E})$

A node v of $\mathcal{G}$ (line 4) is formally defined as a pair containing its single, textual context γ and a finite list ($\mathcal{A}$) of aliases (η_i). γ provides more information about the node and its relations with other nodes while $\mathcal{A}$ are synonymous names or aliases that can be used to refer to the node like identifiers. In Wikidata5m (Wang et al., 2021b), the context of each node is the abstract of the wikipedia page of the entity and each node has aliases similar to Acetaminophen being an alias for Paracetamol. Each (directed) edge e in $\mathcal{G}$ (line 5) is an ordered pair of related nodes where the relation is identified by a set of synonymous aliases $\mathcal{A}$ for the edge (eg. treats, treatment for). Let (v_1, v_2) denote any edge from node v_1 to v_2 in $\mathcal{G}$, where $(v_1, v_2)_{\mathcal{A}}$ denotes the aliases of the edge. Finally, the KG $\mathcal{G}$ (line 8) is a finite collection of nodes $\mathcal{N}$ and edges $\mathcal{E}$.

With the formal structure of the KG defined, the next step is to generate individual reading comprehension questions which serve as samples from a prompt distribution. Each question is constructed based on a directed acyclic subgraph (DAG) of connected facts extracted from the KG. A directed acyclic sub-graph (DAG) $\mathcal{G}' = (\mathcal{N}', \mathcal{E}')$ of $\mathcal{G}$ is a subgraph of $\mathcal{G}$ ($\mathcal{N}' \subseteq \mathcal{N}, \mathcal{E}' \subseteq \mathcal{E}$) consisting of directed edges and with no

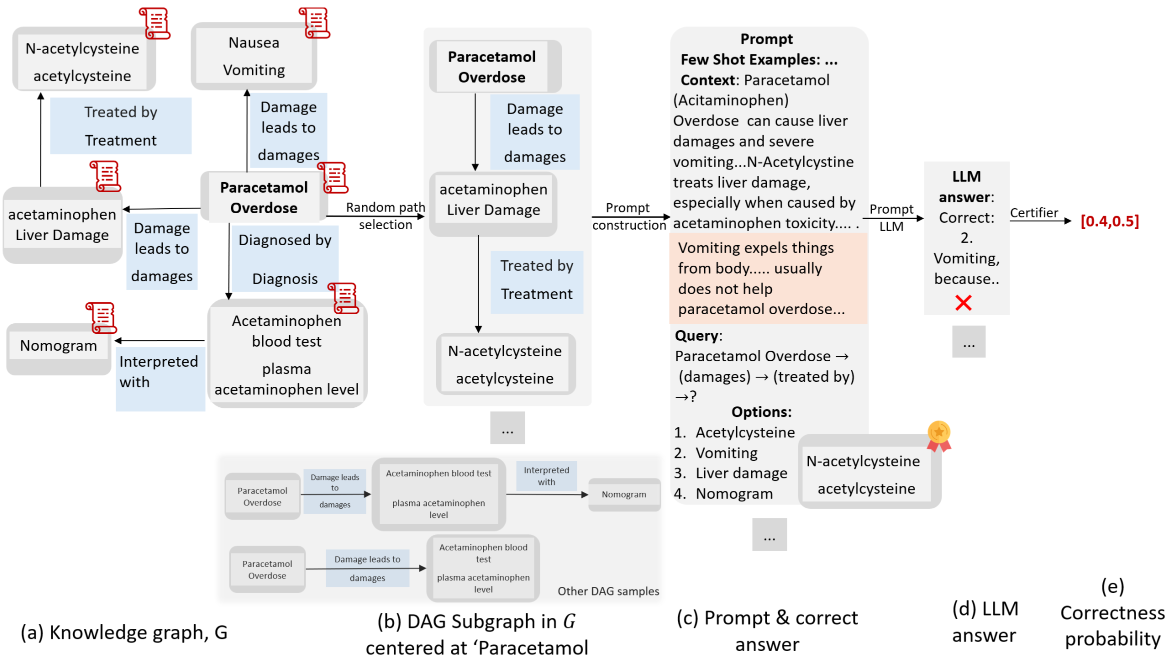


Figure 2: Overview of LLMCert-C. (a) A KG $\mathcal{G}$. (b) A randomly sampled path DAG $\mathcal{G}'$ originating at fixed node 'Paracetamol Overdose', from the various other possible DAGs originating at the node in $\mathcal{G}$. (c) A prompt created from $\mathcal{G}'$ having contexts of the nodes in $\mathcal{G}'$, a distractor context (highlighted in orange, as the node for 'Vomiting' is a distractor for $\mathcal{G}'$), and a query from $\mathcal{G}'$. (d) The target LLM's response to the prompt, validated using the correct answer. (e) Certifier obtains bounds on the probability of correct response using n samples of LLM responses to randomly sampled prompts.

cycles. Let the nodes of $\mathcal{G}'$ be topologically ordered as $\mathcal{N}' = [v_1, v_2 \ldots, v_l]$. Let the i^{th} ($i \in [1, l]$) nodes of $\mathcal{N}'$ from v_1 and backwards from v_l be $\mathcal{N}'[i] \coloneqq v_i$ and $\mathcal{N}'[-i] \coloneqq v_{l-i+1}$ respectively. Nodes $\mathcal{N}'_{source}$ with no incoming edges are source nodes of $\mathcal{G}'$. Nodes $\mathcal{N}'_{sink}$ with no outgoing edges are sink nodes of $\mathcal{G}'$. DAGs generalize reasoning paths that are typically used to define multi-hop reading comprehension queries (Yang et al., 2018; Welbl et al., 2018). Hence, we define generalized multi-hop reading comprehension queries with DAGs in Definition 2.1.

Definition 2.1. (Generalized multi-hop queries). Consider a DAG $\mathcal{G}'$ in $\mathcal{G}$ with nodes $\mathcal{N}'$. A multi-hop query $\mathcal{Q}$ for $\mathcal{N}'$ is to identify the sink nodes $\mathcal{N}'_{sink}$, given aliases of source nodes $\mathcal{N}'_{source}$ and aliases of all edges in DAG $\mathcal{G}'$. To avoid ambiguity regarding correct answer, we consider only the DAGs having a unique sink node, i.e., $\mathcal{N}'_{sink} = \{\mathcal{N}'[-1]\}$.

$\mathcal{G}$ naturally encodes several multi-hop queries, that form the sample space of the specification distributions for a language model $\mathcal{L}$. The final prompts also include relevant textual context needed to identify the intermediate and final nodes to answer the query. The context may contain natural noise, such as distractor texts or jumbled information, that can occur in practical settings. We model such noise in our prompt distributions as described next.

2.1.1 Natural noise in prompts

Distractors. Prior works (Shi et al., 2023) on analyzing reasoning in Language Models (LMs) have shown the negative influence of irrelevant information (distractor) in prompts on LM performance. Hence, we include distractor texts in prompt contexts and specify that model response should not use the distractor information. Contexts of nodes $\tilde{v}$ adjacent to any node $\mathcal{N}[i]$ ($i \in [1, l]$) in $\mathcal{N}$ that are not the sink node or its predecessors, such that the aliases of the edge $(\mathcal{N}[i], \tilde{v})$ are the same as those of $(\mathcal{N}[i], \mathcal{N}[j])$, $j > i$ in the DAG, can serve as effective distractors for $\mathcal{Q}$ (Definition 2.2). This is because, at any intermediate step, the LM $\mathcal{L}$ can pick $\tilde{v}$ as the response, which can deviate $\mathcal{L}$'s reasoning from $\mathcal{N}$. Nodes adjacent to $\mathcal{N}[-1]$ and its predecessors in $\mathcal{N}$ are not distractors. For the former, the LM must have already reached the final answer before reaching its adjacent nodes, hence answering $\mathcal{Q}$. In the latter, adjacent nodes following same relation are valid answers, not distractors.

Definition 2.2. (Distractor). Consider DAG $\mathcal{G}'$ in $\mathcal{G}$, with nodes $\mathcal{N}'$ and $\mathcal{N}$ respectively. A distractor node $\tilde{v} \notin \mathcal{N}'$ satisfies $\exists v \in \mathcal{N} \setminus \mathcal{N}', ((v, \tilde{v}) \in \mathcal{E}) \wedge (\exists v' \in \mathcal{N}', (v, \tilde{v})_{\mathcal{A}} = (v, v')_{\mathcal{A}} \wedge v' \neq \mathcal{N}'[-1])$.

Shuffling. Prior works (Chen et al., 2024) have shown that LM performance can vary with information ordering. Hence, we shuffle the information provided in the prompt, to specify that LM's response should be invariant to information ordering.

Our certificates quantify the probability p of observing correct reading comprehension for a random multi-hop reasoning prompt, with possible natural noise, developed from $\mathcal{G}$. We formally define p for $\mathcal{L}$ as a probabilistic program over $\mathcal{G}$ in Algorithm 1. We follow the syntax of the imperative probabilistic programming language in (Sankaranarayanan et al., 2013, Figure 3). The language has primitives for sampling from common distributions like Uniform ($\mathcal{U}$), Bernoulli (Ber), etc., and `estimateProb(.)` function that outputs the probability of a random variable attaining a certain value. We use a generic identifier $\mathcal{D}$ (line 1) for samplers of discrete distributions ('$\ldots$' denotes samplers for other discrete distributions). We use `any`(.) function to denote that at least 1 of its inputs is true.

A prompt for $\mathcal{L}$ consists of $\mathcal{Q}$ and a context Γ containing information relevant to answer $\mathcal{Q}$ (Algorithm 1, line 6). $\mathcal{Q}$ is developed from a DAG $\mathcal{G}'$, randomly sampled by the function `sampleDAG`. We elaborate on `sampleDAG`() in §2.1.2. $\mathcal{G}'$ is transformed into a natural language query with a randomly-selected template from a set of prespecified templates, τ. The template samples random aliases of nodes in $\mathcal{N}'_{source}$ and relations in $\mathcal{G}'$ to form a natural language query for $\mathcal{N}'_{sink}$. Γ is formed by concatenating ($\odot$) contexts for all nodes in $\mathcal{N}'$, followed by optional shuffling. We denote distractor text in Γ as context of randomly sampled nodes from a distribution $\mathcal{D}$ over all distractor nodes of $\mathcal{N}'$ in $\mathcal{N}$. Our final certificate (line 7) quantifies the probability p that $\mathcal{L}$ generates any alias of $\mathcal{N}'_{sink}$, which is correct answer. The certificate depends on the different sampling steps and is specific to $\mathcal{G}$.

Algorithm 1 General certification

```
Input: $\mathcal{L}, \mathcal{G}, \tau$, args
Output: $p$
1: $\mathcal{D} := \mathcal{U} \mid Ber \mid \ldots$
2: $\mathcal{G}' :=$ sampleDAG($\mathcal{G}$, args)
3: $\mathcal{N}' :=$ topologicalOrder($\mathcal{G}'$)
4: $\mathcal{Q} := \mathcal{D}(\tau)(\mathcal{G}', \mathcal{N}')$
5: $\Gamma :=$ shuffle($[\mathcal{N}'[0]_\gamma, \ldots, \mathcal{N}'[-1]_\gamma, (\mathcal{D}(\mathcal{N}))_\gamma, \ldots]$)
6: $\mathcal{P} := \Gamma \odot \mathcal{Q}$
7: $p :=$ estimateProb(any($\mathcal{L}(\mathcal{P}) == \mathcal{N}'[-1]_\mathcal{A}$))
```

Algorithm 2 Entity centric `sampleDAG`

```
Input: $\mathcal{G}, \rho$; Output: $\mathcal{G}'$
1: $l := \mathcal{D}(\{2, \ldots, \rho\})$
2: $\mathcal{B} :=$ boundedDAGs($\mathcal{G}, \mathcal{N}_{source}, l$)
3: $\mathcal{G}' := \mathcal{D}(\mathcal{B})$
```

Algorithm 3 Relations centric `sampleDAG`

```
Input: $\mathcal{G}, \mathcal{G}_{ref}$; Output: $\mathcal{G}'$
1: $\mathcal{I} :=$ generateIsomorphisms($\mathcal{G}, \mathcal{G}_{ref}$)
2: $\mathcal{G}' := \mathcal{D}(\mathcal{I})$
```

2.1.2 Local specifications

We define individual reading comprehension specifications on queries over DAGs that share a common characteristic, thus forming *local specifications* (Seshia et al., 2018). Local specifications are commonly defined for and used in neural network verification (Singh et al., 2019; Baluta et al., 2021), due to their tractability to verify and simplicity to interpret. Although there can be several common characteristics for DAGs, we define two kinds of local specifications with specific common characteristics. `sampleDAG`() randomly samples DAGs with the common characteristics from $\mathcal{G}$.

Entity centric. Entity-centric question answering (Liu et al., 2023; Sciavolino et al., 2021; Shavarani and Sarkar, 2024) is an important question-answering paradigm, where the focus is on questions related to particular (real-world) entities. It is relevant in various real-world applications such as topic-specific learning (Liu et al., 2003) and assisted reading (Yu et al., 2020) for localizing at information of interest. We specify reading comprehension over entity-centric questions by randomly sampling DAGs starting at a fixed set of nodes $\mathcal{N}_{source}$. From a practical standpoint, queries from DAGs with longer paths between source and sink nodes can become meaningless (e.g., Paul Sophus Epstein$\xrightarrow{\text{place of death}}\xrightarrow{\text{administrative unit}}\xrightarrow{\text{country}}\xrightarrow{\text{capital}}\xrightarrow{\text{name origin}}$ **?**), and thus shorter graphs are considered in popular question-answering datasets such as (Yang et al., 2018; Trivedi et al., 2022). Thus, we upper-bound the size of $\mathcal{N}'$ considered in the specification, by a hyperparameter $\rho \geq 2$. Let `boundedDAGs` be a function that generates all the DAGs in $\mathcal{G}$, $\mathcal{B}$, having the common characteristic that they start from nodes in $\mathcal{N}_{source}$, such that the source and sink nodes have paths of at most $l \in [2, \rho]$ nodes. `sampleDAG` returns a randomly sampled DAG $\mathcal{G}'$ from $\mathcal{B}$ (Algorithm 2).

Relations centric. Question-answering datasets are commonly formed with question templates (Cui et al., 2016) each having fixed relations (relation-centric) between variable entities of specific types. For example, "Which drug is a treatment for \${disease}?", where \${disease} can be substituted with any curable disease to form individual questions. We define specifications by fixing the structure and relations of DAG with a reference DAG $\mathcal{G}_{ref}$ and randomly sampling nodes to obtain the final DAG $\mathcal{G}'$ over which (relation-centric) queries are formed. $\mathcal{G}'$ is an isomorphism (Definition 2.3) of $\mathcal{G}_{ref}$. Algorithm 3 defines the sampling procedure for $\mathcal{G}'$ using $\mathcal{G}_{ref}$.

Definition 2.3. (Isomorphic DAGs) Let finite DAGs $\mathcal{G}_1 = (\mathcal{N}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{N}_2, \mathcal{E}_2)$ be isomorphic when there exists a bijection ϕ between the nodes of $\mathcal{G}_1$ and $\mathcal{G}_2$ such that the following conditions hold. (1) $\forall v_1, v_2 \in \mathcal{N}_1, (v_1, v_2) \in \mathcal{E}_1 \Leftrightarrow (\phi(v_1), \phi(v_2)) \in \mathcal{E}_2$; (2) $\forall (v_1, v_2) \in \mathcal{E}_1, (v_1, v_2)_{\mathcal{A}} = (\phi(v_1), \phi(v_2))_{\mathcal{A}}$. `generateIsomorphisms`() gives all isomorphisms of $\mathcal{G}_{ref}$ from $\mathcal{G}$.

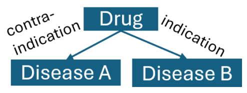
2.2 Certification

LLMCert-C certifies the target LLM $\mathcal{L}$ by computing an interval $[p_l, p_u]$ bounding probability p (Algorithm 1, line 7) for a given specification distribution over $\mathcal{G}$ with high confidence. We model p as the probability of setting the underlying boolean random variable $\mathcal{R} \triangleq (\texttt{any}(\mathcal{L}(\mathcal{P}) == \mathcal{N}'[-1]_{\mathcal{A}}))$ to true (success). Thus, $\mathcal{R} \sim Ber(p)$. Exactly determining p would require enumerating over all possible $\mathcal{P}$ which can be developed from any DAG of $\mathcal{G}$ with any random aliases, resulting in an infeasible number of possible prompts, as shown in Appendix 8.8. Moreover, we want our method to generalize to closed-source LLMs as well, where the internal structures of the models are unknown. Hence, we cannot use any symbolic methods (Mirman et al., 2020) to determine p. Thus, to scalably certify the black-box target LM $\mathcal{L}$, we estimate p with provably high-confidence (low error) bounds. Confidence is defined as the probability of the true value p being within the bounds, i.e., $Pr[p \in [p_l, p_u]]$. To establish formal guarantees, we want our estimation procedure to be such that the actual confidence is at least the user-specified confidence level, $1-\delta$ (i.e., $Pr[p_l \leq p \leq p_u] \geq 1-\delta$), where $\delta > 0$ is a small constant. Hence, we use Clopper-Pearson (CP) confidence intervals (Clopper and Pearson, 1934; Kurz et al., 2014), which is a conservative method known to produce guaranteed high confidence intervals. We show evidence of the bounds' conservativeness in Appendix 7. To compute high-confidence bounds on p, we make n independent and identically distributed observations of $\mathcal{R}$, in which we obtain k successes, $k \in [0, n]$. We generate CP confidence intervals on p with the n observations and $1-\delta$ confidence. The time complexity of certification is linear in n.

3 Experiments

Our experiments were run on 2 A100 GPUs with 40GB VRAM each. We certify following open-source models — Llama-3.2-instruct 1B, 3B, 11B (Dubey et al., 2024), Phi-3 3B and 14B (Abdin et al., 2024) and their 4-bit quantizations to study the effects of quantization on reading comprehension. Among closed-source models with API access, we certify Gemini-1.5 (Gemini Team, 2024) Flash-001 and Pro-002 and GPT-4o-0827 (OpenAI, 2024). We use PrimeKG (Chandak et al., 2023) and Wikidata5m (Wang et al., 2021b) as our knowledge graphs (KGs) (details in Appendix 8.3.1 and Appendix 8.2.1 respectively). PrimeKG is for precision medicine, while Wikidata5m is for general knowledge from Wikipedia. These practical KGs are chosen only for illustration, and our framework generalizes to other KGs. We show relation-centric and entity-centric specifications for PrimeKG and Wikidata5m, respectively.

PrimeKG. We manually define 9 relation-centric specifications based on common clinical reasoning patterns. All specifications are shown in §8.4.2. Each specification is defined by: 1. A reference DAG ($\mathcal{G}_{ref}$); 2. A set of natural language templates (5-7 per specification); 3. Constraints on node entity types for isomorphic DAG sampling. An example is the "drug-disease interaction" specification. It consists of $\mathcal{G}_{ref}$ which is a 3-node DAG (shown on the right) with drug→disease edges. An example template to form natural language queries from $\mathcal{G}_{ref}$ is "Which drug treats disease B but is contraindicated for disease A?".



Wikidata5m. We created 50 entity-centric specifications using Wikidata5m, sampling linear paths (DAGs) up to length $\rho = 5$ to maintain query relevance. The 50 starting nodes were selected from high out-degree (top 2k) or high-density neighborhood (>2k nodes within radius ρ) populations so that each of the specifications has a large number of paths, making enumeration infeasible. Note that these specifications are only for demonstration and LLMCert-C is not limited to them. To avoid bias towards the naturally more frequent shorter paths, we first sample a desired path length uniformly from $[2, \rho]$. A path of the selected length is then generated from the starting node (Algorithm 2). We ablate on query path lengths in Appendix 9. Note that the framework can be instantiated with other distributions specific to the usecase.

Specifications with varying complexities. We certify 2 kinds of specifications with varying contexts in LLM prompts (Algorithm 1, line 4) — with context shuffling and distractor context (Distractor) and without such noise (Vanilla). This enables us to study the effects of natural noise on LLM performance. Leveraging the linear structure of Wikidata5m DAGs, we use weighted sampling (Algorithm 6) to select distractors closer to the answer node, hypothesizing this increases difficulty. Since PrimeKG DAGs lack such linearization, its distractor nodes are sampled uniformly. We

Table 1: Average of LLM certification bounds [avg of lower bounds, avg of upper bounds] for Wikidata5m and PrimeKG specifications.

Model	Quantization	Wikidata			PrimeKG		
		Baseline	Vanilla	Distractor	Baseline	Vanilla	Distractor
Gemini-1.5-Pro	-	0.87	[0.80, 0.89]	[0.64, 0.75]	1.00	[0.97, 1.00]	[0.92, 0.98]
GPT-4o	-	0.88	[0.80, 0.89]	[0.62, 0.74]	0.99	[0.97, 1.00]	[0.85, 0.95]
Gemini-1.5-Flash	-	0.81	[0.72, 0.83]	[0.43, 0.56]	0.99	[0.96, 0.99]	[0.89, 0.96]
Phi-3 (14B)	fp16	0.63	[0.54, 0.66]	[0.33, 0.45]	0.98	[0.93, 0.98]	[0.75, 0.85]
	4bit	0.63	[0.52, 0.65]	[0.30, 0.42]	0.95	[0.92, 0.97]	[0.71, 0.81]
Llama (11B)	fp16	0.61	[0.50, 0.63]	[0.33, 0.45]	0.97	[0.91, 0.97]	[0.84, 0.92]
	4bit	0.54	[0.45, 0.57]	[0.29, 0.41]	0.94	[0.88, 0.95]	[0.79, 0.88]
Phi-3 (3B)	fp16	0.58	[0.50, 0.61]	[0.32, 0.45]	0.92	[0.85, 0.92]	[0.70, 0.81]
	4bit	0.57	[0.48, 0.60]	[0.29, 0.41]	0.89	[0.83, 0.91]	[0.74, 0.83]
Llama (3B)	fp16	0.51	[0.43, 0.55]	[0.27, 0.39]	0.87	[0.80, 0.89]	[0.70, 0.80]
	4bit	0.46	[0.39, 0.51]	[0.24, 0.35]	0.84	[0.73, 0.83]	[0.63, 0.73]
Llama (1B)	fp16	0.38	[0.30, 0.41]	[0.22, 0.33]	0.48	[0.43, 0.56]	[0.39, 0.52]
	4bit	0.34	[0.27, 0.38]	[0.20, 0.31]	0.38	[0.33, 0.45]	[0.28, 0.40]

provide an ablation study on the effects of varying the weighted distractor sampling in Appendix 9. Each prompt is a multiple-choice question with one correct answer, allowing for automated evaluation via string matching. A template (shown below) assembles the final prompt, which includes few-shot examples, the query, answer options, and a context section. For the Distractor setting, this context is a mixture of relevant and distracting information. For the multiple-choice options in distractor setting, we prioritize distractor nodes, followed by nodes from the query path, and finally, random nodes related to path nodes. Vanilla setting is similar but contains no distractors. Complete details of our prompt construction are in Appendix 8.5.

Prompt Template

{few_shot_examples}
Given Context: {context} Answer the question: {query} by selecting the correct answer from the following options: {options}
The format for beginning your response is:
correct answer: ⟨*option_number*⟩.⟨*answer*⟩, because ⟨*succinct reason*⟩
follow this format and only choose from the options

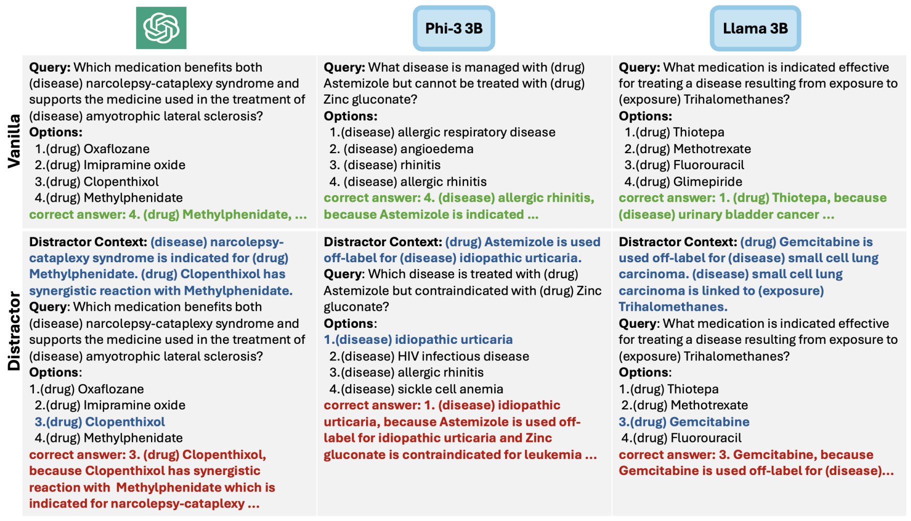
Baseline. We compare LLMCert-C's results with a benchmarking baseline. It measures accuracy on a static dataset. We generate 50 static queries from each specification by uniformly sampling DAGs. Queries use a fixed alias of entities and relations, with no shuffling or distractors in contexts to remove natural noise.

3.1 Certificates

LLMCert-C generates certificates with 95% confidence that are tight lower and upper bounds on the probability of a correct LLM response to a random prompt sampled from the specified prompt distributions. LLMCert-C uses 250 samples. We chose this sample size using an ablation study (Appendix 9.5), which shows that the stability of the certified bounds sees diminishing returns beyond 200 samples, making 250 a sufficient and efficient choice. We report the average value of the lower and upper bounds respectively, over all specifications over a KG that LLMCert-C certifies for each LLM, in Table 1. Each certificate takes 10-20 minutes for generation, where the main bottleneck is LLM inference. We plan to optimize this in future work. We also report the average baseline result for each static dataset from the specifications. Appendix 9 presents detailed ablation studies for the impact of the various certification hyperparameters. Next, we summarize the key observations from Table 1.

Variation in reading comprehension by models. We observe that larger models, like Gemini and GPT, have significantly higher bounds than those for smaller models, like Phi-3, Llama, for specifications over both KGs. The lower bounds of the larger models are higher than the upper bounds of the smaller models, suggesting a paradigm shift in reading comprehension capabilities (especially for Gemini-1.5-Pro and GPT-4o). However, as the larger models are closed-source, we are unaware whether the enhanced reading comprehension capabilities could be due to specialized training or fine-tuning techniques applied on the models. We can establish model performance hierarchies using the certificates with high confidence. Specifically, higher lower bound of a model than the upper bound of another model (e.g., Gemini-v1.5-Pro vs Llama 11B for all settings), indicates superior performance of the former in the worst case with high confidence.

Effects of different kinds of specifications. Our results indicate that Vanilla specifications are easier and lead to higher certification bounds. Distractor



		Phi-3 3B	Llama 3B
Vanilla	**Query:** Which medication benefits both (disease) narcolepsy-cataplexy syndrome and supports the medicine used in the treatment of (disease) amyotrophic lateral sclerosis? **Options:** 1.(drug) Oxaflozane 2.(drug) Imipramine oxide 3.(drug) Clopenthixol 4.(drug) Methylphenidate **correct answer: 4. (drug) Methylphenidate, ...**	**Query:** What disease is managed with (drug) Astemizole but cannot be treated with (drug) Zinc gluconate? **Options:** 1.(disease) allergic respiratory disease 2. (disease) angioedema 3. (disease) rhinitis 4. (disease) allergic rhinitis **correct answer: 4. (disease) allergic rhinitis, because Astemizole is indicated ...**	**Query:** What medication is indicated effective for treating a disease resulting from exposure to (exposure) Trihalomethanes? **Options:** 1.(drug) Thiotepa 2.(drug) Methotrexate 3.(drug) Fluorouracil 4.(drug) Glimepiride **correct answer: 1. (drug) Thiotepa, because (disease) urinary bladder cancer ...**
Distractor	**Distractor Context: (disease) narcolepsy-cataplexy syndrome is indicated for (drug) Methylphenidate. (drug) Clopenthixol has synergistic reaction with Methylphenidate.** **Query**: Which medication benefits both (disease) narcolepsy-cataplexy syndrome and supports the medicine used in the treatment of (disease) amyotrophic lateral sclerosis? **Options**: 1.(drug) Oxaflozane 2.(drug) Imipramine oxide **3.(drug) Clopenthixol** 4.(drug) Methylphenidate **correct answer: 3. (drug) Clopenthixol, because Clopenthixol has synergistic reaction with Methylphenidate which is indicated for narcolepsy-cataplexy ...**	**Distractor Context: (drug) Astemizole is used off-label for (disease) idiopathic urticaria.** **Query**: Which disease is treated with (drug) Astemizole but contraindicated with (drug) Zinc gluconate? **Options**: **1.(disease) idiopathic urticaria** 2.(disease) HIV infectious disease 3.(disease) allergic rhinitis 4.(disease) sickle cell anemia **correct answer: 1. (disease) idiopathic urticaria, because Astemizole is used off-label for idiopathic urticaria and Zinc gluconate is contraindicated for leukemia ...**	**Distractor Context: (drug) Gemcitabine is used off-label for (disease) small cell lung carcinoma. (disease) small cell lung carcinoma is linked to (exposure) Trihalomethanes.** **Query**: What medication is indicated effective for treating a disease resulting from exposure to (exposure) Trihalomethanes? **Options**: 1.(drug) Thiotepa 2.(drug) Methotrexate **3.(drug) Gemcitabine** 4.(drug) Fluorouracil **correct answer: 3. Gemcitabine, because Gemcitabine is used off-label for (disease)...**

Figure 3: Negative effects of distractors on LLM performance for answering queries from same DAGs (correct and wrong answers in green and red respectively).

specifications are challenging for all models, leading to upper bounds that are consistently lower than the lower bounds for the Vanilla setting. This suggests *susceptibility of SOTA LLMs to natural noise in prompts*. Even larger models such as Gemini-1.5-Pro, which show close to perfect performance in the baseline and Vanilla case (for PrimeKG only), can be incorrect with distractors, showing the nontrivial effects of noise.

Comparison with benchmarking baseline. Baseline scores of all models approach or surpass average certification upper bounds, suggesting inflated performance estimates in benchmarking. For example, contrary to certification bounds for PrimeKG's distractor specifications, Phi-3 (14B) outperformed Llama (11B). Such findings emphasize the need for more reliable evaluation methods grounded with statistical guarantees. Contrary to benchmarking, certificates prevent incorrect comparisons between models with similar performance (e.g., Phi-3 14B and Llama 11B and respectively between their 4-bit models for Vanilla specifications) by showing overlaps in their certification bounds.

Effects of model quantization. We see that higher quantization deteriorates model performance on reading comprehension, contrary to prior works like Jin et al. (2024) that suggest that 4-bit quantization can retain the model's knowledge and reasoning capabilities.

3.2 Case studies

We qualitatively analyze certificates for PrimeKG specifications. We discuss similar case studies for Wikidata5m specifications in Appendix 10. First, we identify the vulnerabilities of SOTA LLMs to distractor texts in Figure 3. We see consistent occurrences of incorrect model responses with distractors and not otherwise. This demonstrates the vulnerabilities of SOTA LLMs and the risks posed by their deployment in critical applications. Additionally, we show the responses of 3 models in Appendix 11, Figure 12 — Phi-3 (3B), GPT-4o, and Gemini-Pro, obtained when certifying them for the Distractor specification. The samples reflect the certification bounds, showing the performance hierarchy across the models. Certificates can be used to select the best models for critical applications.

4 Related works

In-context learning. As LLM context windows increase (Gemini Team, 2024; Chen et al., 2023; Dubey et al., 2024), more information can be provided in the prompts (Brown et al., 2020; Qu et al., 2024). In-context learning is the emergent behavior (Wei et al., 2022; Lu et al., 2024) in which LLMs become proficient at a task with demonstrations in prompts. We use in-context learning and few-shot examples to enhance LLMs' reading comprehension.

Benchmarking LLMs. Several benchmarks have been proposed to study reasoning (Huang and Chang, 2023; Plaat et al., 2024; He et al., 2024; Zha et al., 2021), arithmetic (Yuan et al., 2023; Song et al., 2024), planning (Pallagani et al., 2023; Valmeekam et al., 2023), and question-answering (Yang et al., 2018; Welbl et al., 2018; Perevalov et al., 2022) in LLMs. These benchmarks provide empirical insights and trends of LM performance. However, they use static datasets and not guaranteed to generalize. Certification provides guarantees on the scope (defined by specifications) and confidence of the claims, as shown in this work.

Acknowledgement

This work was supported by a grant from the Amazon-Illinois Center on AI for Interactive Conversational Experiences (AICE) and NSF Grants No. CCF-2238079, CCF-2316233, CNS-2148583, NAIRR240476.

References

Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A. A., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., Benhaim, A., and et al (2024). Phi-3 technical report: A highly capable language model locally on your phone.

Baluta, T., Chua, Z. L., Meel, K. S., and Saxena, P. (2021). Scalable quantitative verification for deep neural networks.

Bloom, B. S. (1956). *Taxonomy of Educational Objectives, Handbook: The Cognitive Domain*. David McKay, New York.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., and Liang, P. (2022). On the opportunities and risks of foundation models.

Bonaert, G., Dimitrov, D. I., Baader, M., and Vechev, M. (2021). Fast and precise certification of transformers. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, page 466–481, New York, NY, USA. Association for Computing Machinery.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

Chandak, P., Huang, K., and Zitnik, M. (2023). Building a knowledge graph to enable precision medicine. *Nature Scientific Data*.

Chaudhary, I., Hu, Q., Kumar, M., Ziyadi, M., Gupta, R., and Singh, G. (2024). Quantitative certification of bias in large language models.

Chen, S., Wong, S., Chen, L., and Tian, Y. (2023). Extending context window of large language models via positional interpolation.

Chen, W., Zha, H., Chen, Z., Xiong, W., Wang, H., and Wang, W. (2021). Hybridqa: A dataset of multi-hop question answering over tabular and textual data.

Chen, X., Chi, R. A., Wang, X., and Zhou, D. (2024). Premise order matters in reasoning with large language models.

Clopper, C. J. and Pearson, E. S. (1934). The Use Of Confidence Or Fiducial Limits Illustrated In The Case Of The Binomial. *Biometrika*, 26(4):404–413.

Cui, W., Xiao, Y., and Wang, W. (2016). Kbqa: an online template based question answering system over freebase. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 4240–4241. AAAI Press.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., and et al (2024). The llama 3 herd of models.

Educational Testing Service (2024). Toefl ibt reading section. https://www.ets.org/toefl/test-takers/ibt/about/content/reading.html. Accessed: 2024-09-29.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. (2018). Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18.

Gemini Team, G. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.

Harrison, R. M. (2024). A comparison of large language model and human performance on random number generation tasks.

He, X., Tian, Y., Sun, Y., Chawla, N. V., Laurent, T., LeCun, Y., Bresson, X., and Hooi, B. (2024). G-retriever: Retrieval-augmented generation for textual graph understanding and question answering.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). Measuring massive multitask language understanding.

Huang, J. and Chang, K. C.-C. (2023). Towards reasoning in large language models: A survey.

IDP IELTS (2024). Ielts reading test. https://ielts.idp.com/uae/prepare/article-ielts-reading-test. Accessed: 2024-09-29.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S. (2022). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

Jin, R., Du, J., Huang, W., Liu, W., Luan, J., Wang, B., and Xiong, D. (2024). A comprehensive evaluation of quantization strategies for large language models.

Kurz, D., Lewitschnig, H., and Pilz, J. (2014). Decision-theoretical model for failures which are tackled by countermeasures. *IEEE Transactions on Reliability*, 63(2):583–592.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., Newman, B., Yuan, B., Yan, B., Zhang, C., Cosgrove, C., Manning, C. D., Ré, C., Acosta-Navas, D., Hudson, D. A., Zelikman, E., Durmus, E., Ladhak, F., Rong, F., Ren, H., Yao, H., Wang, J., Santhanam, K., Orr, L., Zheng, L., Yuksekgonul, M., Suzgun, M., Kim, N., Guha, N., Chatterji, N., Khattab, O., Henderson, P., Huang, Q., Chi, R., Xie, S. M., Santurkar, S., Ganguli, S., Hashimoto, T., Icard, T., Zhang, T., Chaudhary, V., Wang, W., Li, X., Mai, Y., Zhang, Y., and Koreeda, Y. (2023). Holistic evaluation of language models.

Liu, B., Chin, C. W., and Ng, H. T. (2003). Mining topic-specific concepts and definitions on the web. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, page 251–260, New York, NY, USA. Association for Computing Machinery.

Liu, Y., Cao, J., Liu, C., Ding, K., and Jin, L. (2024). Datasets for large language models: A comprehensive survey.

Liu, Y., Huang, J., and Chang, K. (2023). Ask to the point: Open-domain entity-centric question generation. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2703–2716, Singapore. Association for Computational Linguistics.

Lu, S., Bigoulaeva, I., Sachdeva, R., Madabushi, H. T., and Gurevych, I. (2024). Are emergent abilities in large language models just in-context learning?

Mirman, M., Gehr, T., and Vechev, M. (2020). Robustness certification of generative models.

Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., and Farajtabar, M. (2024). Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models.

National Center for Education Statistics (2024). Reading — naep. Technical report, U.S. Department of Education. Accessed: 2024-09-29.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., Das-Sarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. (2022). In-context learning and induction heads.

OpenAI (2024). Hello GPT-4o.

Pallagani, V., Muppasani, B., Murugesan, K., Rossi, F., Srivastava, B., Horesh, L., Fabiano, F., and Loreggia, A. (2023). Understanding the capabilities of large language models for automated planning.

Pang, R. Y., Parrish, A., Joshi, N., Nangia, N., Phang, J., Chen, A., Padmakumar, V., Ma, J., Thompson, J., He, H., and Bowman, S. R. (2022). Quality: Question answering with long input texts, yes!

Peng, C., Xia, F., Naseriparsa, M., and Osborne, F. (2023). Knowledge graphs: Opportunities and challenges.

Perevalov, A., Yan, X., Kovriguina, L., Jiang, L., Both, A., and Usbeck, R. (2022). Knowledge graph question answering leaderboard: A community resource to prevent a replication crisis. In Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Odijk, J., and Piperidis, S., editors, *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2998–3007, Marseille, France. European Language Resources Association.

Plaat, A., Wong, A., Verberne, S., Broekens, J., van Stein, N., and Back, T. (2024). Reasoning with large language models, a survey.

Qu, G., Li, J., Li, B., Qin, B., Huo, N., Ma, C., and Cheng, R. (2024). Before generation, align it! a novel and effective strategy for mitigating hallucinations in text-to-sql generation.

Quach, V., Fisch, A., Schuster, T., Yala, A., Sohn, J. H., Jaakkola, T., and Barzilay, R. (2024). Conformal language modeling. In *The Twelfth International Conference on Learning Representations*.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. (2023). Gpqa: A graduate-level google-proof q&a benchmark.

Sankaranarayanan, S., Chakarov, A., and Gulwani, S. (2013). Static analysis for probabilistic programs: inferring whole program properties from finitely many paths. *SIGPLAN Not.*, 48(6):447–458.

Sciavolino, C., Zhong, Z., Lee, J., and Chen, D. (2021). Simple entity-centric questions challenge dense retrievers. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6138–6148, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Seshia, S. A., Desai, A., Dreossi, T., Fremont, D. J., Ghosh, S., Kim, E., Shivakumar, S., Vazquez-Chanlatte, M., and Yue, X. (2018). Formal specification for deep neural networks. In Lahiri, S. K. and Wang, C., editors, *Automated Technology for Verification and Analysis*, pages 20–34, Cham. Springer International Publishing.

Shavarani, H. S. and Sarkar, A. (2024). Entity retrieval for answering entity-centric questions.

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E., Schärli, N., and Zhou, D. (2023). Large language models can be easily distracted by irrelevant context.

Shi, Z., Wei, J., Xu, Z., and Liang, Y. (2024). Why larger language models do in-context learning differently?

Shi, Z., Zhang, H., Chang, K.-W., Huang, M., and Hsieh, C.-J. (2020). Robustness verification for transformers.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. (2019). An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL).

Song, P., Yang, K., and Anandkumar, A. (2024). Towards large language models as copilots for theorem proving in lean.

Street, W., Siy, J. O., Keeling, G., Baranes, A., Barnett, B., McKibben, M., Kanyere, T., Lentz, A., y Arcas, B. A., and Dunbar, R. I. M. (2024). Llms achieve adult human performance on higher-order theory of mind tasks.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models.

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. (2022). MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*.

Valmeekam, K., Sreedharan, S., Marquez, M., Olmo, A., and Kambhampati, S. (2023). On the planning abilities of large language models (a critical investigation with a proposed benchmark).

Vega, J., Chaudhary, I., Xu, C., and Singh, G. (2023). Bypassing the safety training of open-source llms with priming attacks.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C., and Kolter, J. Z. (2021a). Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages 29909–29921.

Wang, X., Gao, T., Zhu, Z., Zhang, Z., Liu, Z., Li, J., and Tang, J. (2021b). Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022). Emergent abilities of large language models.

Wei, J., Wei, J., Tay, Y., Tran, D., Webson, A., Lu, Y., Chen, X., Liu, H., Huang, D., Zhou, D., and Ma, T. (2023). Larger language models do in-context learning differently.

Welbl, J., Stenetorp, P., and Riedel, S. (2018). Constructing datasets for multi-hop reading comprehension across documents.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., Yin, Z., Dou, S., Weng, R., Cheng, W., Zhang, Q., Qin, W., Zheng, Y., Qiu, X., Huang, X., and Gui, T. (2023). The rise and potential of large language model based agents: A survey.

Xu, X., Kong, K., Liu, N., Cui, L., Wang, D., Zhang, J., and Kankanhalli, M. (2024). An LLM can fool itself: A prompt-based adversarial attack. In *The Twelfth International Conference on Learning Representations*.

Yang, X., Li, Y., Zhang, X., Chen, H., and Cheng, W. (2023). Exploring the limits of chatgpt for query or aspect-based text summarization.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. (2018). HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yu, Q., Bing, L., Zhang, Q., Lam, W., and Si, L. (2020). Review-based question generation with adaptive instance transfer and augmentation. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 280–290, Online. Association for Computational Linguistics.

Yuan, Z., Jiang, H., Zhu, Z., Gao, C., Xia, Y., Jiang, Y., Dong, Y., Zhao, J., and Zhu, W. (2023). How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*.

Zha, H., Chen, Z., and Yan, X. (2021). Inductive relation prediction by bert.

5 A note for practitioners

Our work takes the first step towards providing formal guarantees of LLMs' knowledge comprehension capabilities, with particular relevance for high-stakes domains like healthcare. While our framework enables more reliable LLM deployment through rigorous guarantees, these certificates should be viewed as probabilistic bounds rather than absolute guarantees of model reliability. We anticipate this work will help make LLMs more trustworthy for safety-critical applications, although continued human supervision remains essential. Future work can extend our framework to certify additional properties like safety and robustness, and integrate it with knowledge graph construction methods to handle less structured inputs. With a one-time effort to construct domain-specific knowledge graphs, practitioners can obtain reliable insights into the performance of LLMs for question-answering and knowledge comprehension tasks. We release our implementation to support further research on formal verification of language models.

6 Background

6.1 Knowledge graph

A knowledge graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is a collection of nodes $\mathcal{N}$ representing entities, interconnected by directed edges $\mathcal{E}$ representing their relations (Peng et al., 2023; Ji et al., 2022). They are commonly used in search engines to improve the accuracy of responses to user queries. Hence, big companies develop their own closed-source knowledge graphs. PrimeKG (Chandak et al., 2023) is an open-source knowledge graph of precision medicine consisting of nodes corresponding to diseases, drugs, effects, etc. PrimeKG describes ∼17k diseases with ∼4M relations. Wikidata5m (Wang et al., 2021b), another popular open-source knowledge graph consisting of 5M nodes, is a structured representation of Wikipedia. Each Wikidata node corresponds to a Wikipedia page, containing its abstract and a set of aliases that can synonymously refer to the node. Two nodes (v_1, v_2), $v_1, v_2 \in \mathcal{N}$ are connected by a labeled edge if there is a link in the supporting document for v_1 to that for v_2. Edge (v_1, v_2) is labeled by a set of aliases for the relation between v_1 and v_2. App. Figure 4 shows a subgraph of Wikidata5m.

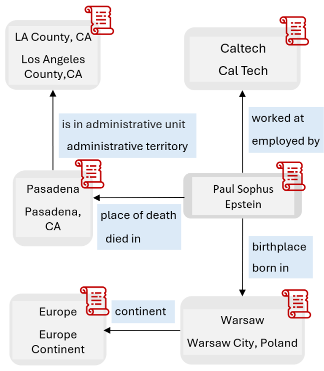


Figure 4: A subgraph of Wikidata5m

6.2 Large Language Models

Large Language Models (LLMs) are autoregressive causal language models that operate on a vocabulary $\mathcal{V}$, a set of tokens. LLMs takes a sequence of tokens $x_1, ..., x_n$ where $x_i \in \mathcal{V}, n > 0$ and outputs a probability distribution over $\mathcal{V}$ to sample the next token x_{n+1}. These models are pretrained on vast corpora (Liu et al., 2024) and have shown remarkable capabilities (Touvron et al., 2023; Gemini Team, 2024; OpenAI, 2024). Numerous benchmarks (Yang et al., 2018; Rein et al., 2023; Hendrycks et al., 2021) have been developed to evaluate the performance of LLMs on tasks related to multi-step reasoning, knowledge comprehension and question answering. However, there remains a gap in our theoretical understanding of LLMs' capabilities.

7 Validity of Certification bounds

In this section, we provide empirical evidence for the validity of the certification bounds with the user-defined confidence (95%). We select 10 equally-spaced values of the true probability of correct LLM response, between 0 and 1. For each true probability, we generate 1000 Clopper-Pearson confidence intervals, which are used to form our certification bounds. We calculate the proportion of the confidence intervals that contain the true probability and compare with the confidence level. We show in Figure 5 that the proportion of instances where the confidence intervals contain the true probability is almost always more than 95%.

8 Knowledge Graph and Query Generation

This section details our experimental setup for generating multi-hop reasoning queries using the Wikidata5m knowledge graph. We describe the structure of the knowledge graph, the process of generating random

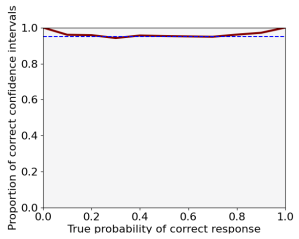


Figure 5: Variation in the proportion of Clopper-Pearson confidence intervals containing the true probability of correct response

paths, formulating queries, and creating answer options including distractors.

8.1 Knowledge Graph Structure

Our experiments utilize two knowledge graphs: Wikidata5m and PrimeKG, each with distinct characteristics.

8.2 Wikidata5m

Wikidata5m is characterized by rich textual and semantic features:

- **Nodes**: Each node represents an entity and is accompanied by a descriptive text paragraph that provides contextual information about the entity.
- **Edges**: Relationships between entities are represented as edges.
- **Text Paragraphs**: Each node contains an associated text paragraph that that often provides at minimum information relevant to its relationships with other entities.
- **Aliases**: Each node and each edge has a set of aliases associated with them which are just different names for them.

8.2.1 Preprocessing the wikidata5m knowledge graph

To ensure the generation of unambiguous queries and support the certification process, we preprocess the wikidata5m dataset.

1. **Relation Filtering:** We remove relations such as 'instance of', 'subclass of', and 'part of' due to their inherent potential for ambiguity in query formulation.
2. **Relevant Information Extraction for edges:** To ensure the relevance of relationships in the knowledge graph, we require textual evidence for each edge. When entity A is related to entity B, we identify specific sentences in the descriptive text of either entity that explicitly mention any alias of the other entity. We assume these sentences support the relationship's existence. These sentences are then linked to the edge, providing context that can be used to answer queries about the relationship. This approach ensures that the knowledge graph contains valid relationships and the specific text that justifies them, enhancing the available context for further analysis. If we find no supporting text for an edge, we drop that edge from the knowledge graph.
3. **Unicode to ASCII:**For consistency within our experiments, we convert all text containing Unicode characters into their respective ASCII approximations.

8.3 PrimeKG

PrimeKG presents a more streamlined structure:

- **Nodes**: The graph consists of nodes representing entities, but unlike Wikidata5m, nodes do not contain accompanying descriptive text.
- **Edges**: Similar to Wikidata5m, edges denote relationships between entities.
- **Aliases**: Each node and edge is associated with exactly one alias, providing a single alternative name rather than multiple variants.

8.3.1 Preprocessing the PrimeKG knowledge graph

The PrimeKG dataset requires several key transformations to align with our experimental framework.

1. **Converting to a Directed Graph:** We convert PrimeKG's undirected structure to a directed graph by creating bidirectional edges. Each original edge generates two directed edges with labels incorporating the target entity's type (e.g., an undirected indication edge between a drug and disease yields "indication (drug)" and "indication (disease)" directed edges).
2. **Enriching Entity Representations:**We augment entity representations by prepending type information to each entity's alias, maintaining a single alias per node to preserve medical precision.
3. **Generating Supporting Text:** To address PrimeKG's lack of contextual information, we generate descriptive text for each edge using templates

tailored to the entity types and relationships (e.g., "entity_1 is relation name for entity_2"). Implementation details are available in the accompanying code.

4. **Standardizing Character Encoding:** All text is converted to ASCII format, matching the Wikidata5m preprocessing pipeline.

8.4 Query Generation

Query generation follows a two-phase process: (1) DAG generation according to the knowledge graph structure and specifications, and (2) transformation of the DAG into a natural language query. The specifics differ between Wikidata5m and PrimeKG due to their distinct structures and certification requirements.

8.4.1 Wikidata5m

For Wikidata5m, we generate linear DAGs (paths) from specified pivot nodes. This approach enables focused testing of an LLM's knowledge comprehension capabilities through a chain of relationships. In this case therefore $\mathcal{N}_{source}$ has only one node the pivot node, v_0. Then we simulate the algorithm 2 and sample a path for our query generation.

From the pivot node v_0, we construct a local subgraph $\mathcal{G}(v_0)$ containing all paths $\mathcal{N}_{v_0}$ originating from v_0. This local subgraph construction serves two purposes: efficient path sampling and enforcement of the ρ constraint from Algorithm 1. We set $\rho = 5$ as longer paths often lose semantic coherence in knowledge comprehension queries.

Within $\mathcal{G}(v_0)$, we generate paths through a randomized sampling process:

1. Sample path length k_{choice} uniformly from $\{1, \ldots, \rho\}$, following line 1 in Algorithm 2

2. Generate path through iterative node sampling: - Start with $\mathcal{N} = [v_0]$ - At each step i, sample next node according to distribution $\mathcal{D}$ (defined in line 1):

$$v_{i+1} \sim \mathcal{D}([v' \mid (v_i, v') \in \mathcal{E} \wedge v' \in \mathcal{G}(v_0)])$$

where v_i is the current terminal node of $\mathcal{N}$

To ensure well-defined queries per Definition 2.1, we require path uniqueness: following the relation sequence of $\mathcal{N}$ from any node must lead to at most one target node. This constraint ensures $|\mathcal{N}'_{sink}| = 1$ as specified in Definition 2.1, while preserving the natural graph structure where nodes may have multiple edges with the same relation to non-path nodes. Additionally, this approach prevents queries with multiple valid answers, which would complicate the evaluation of the language model's performance. It's important to note that this uniqueness constraint applies only to the specific path being generated. Nodes within the path may still have multiple edges with the same relation type to other entities not on the path. This allowance maintains the natural complexity of the knowledge graph structure, where entities can have multiple relationships of the same type with different entities.

The complete path generation process is detailed in Algorithm 4, which implements the path sampling component of `sampleDAG` used in lines 2-3 of Algorithm 2.

The pseudocode for the path generation algorithm is specified in 4

Algorithm 4 Random Path Generation

```
1: Input: Graph $G$, Integer $k$, Vertex $source$
2: Output: $path$
3: $path\_len \leftarrow$ RandomInteger(1, $k$)
4: $path \leftarrow$ None
5: while $path$ is None do
6:     $path \leftarrow$ DFSPath($G$, $source$, $path\_len$)
7:     if not IsUnique($path$) then
8:         $path \leftarrow$ None
9:     end if
10: end while
11: Return $path$
```

Query Formulation: Once a valid path $\mathcal{N}$ is generated, we convert it into a query string. This process aligns with line 3 in Algorithm 1. The query is constructed by sampling aliases for each node and relation in the path. For example, a path $\mathcal{N} = [A, B, C]$ might be converted to a query "sampled_alias(A) $\rightarrow$ sampled_alias((A, B)) $\rightarrow$ sampled_alias(B) $\rightarrow$ sampled_alias((B, C)) $\rightarrow$?". Here the tuple of two nodes represents their edge. The aliases are sampled randomly from a discrete uniform distribution over the available aliases for a node or an edge.

Example Query Generation To illustrate our query generation process, consider the scenario of a path in our subgraph as shown in 6.

Our algorithm would construct the following query from the path presented in 6:

"Chandler Bing$\rightarrow$(actor)$\rightarrow$(birth_date)$\rightarrow$?"

This query requires the LLM to reason through two hops in the knowledge graph:

1. Identify the actor who played Chandler Bing (Matthew Perry)

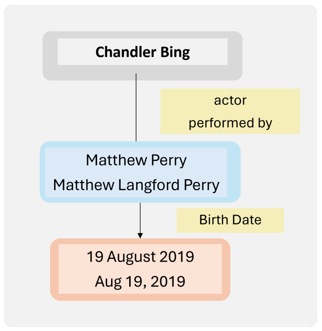


Figure 6: Potential Path in a Subgraph where Pivot is Chandler Bing

2. Find the birth date of Matthew Perry (19 August 1969)

This example demonstrates how our query generation process creates questions that require multi-hop reasoning, leveraging the structure and relationships within the knowledge graph.

8.4.2 PrimeKG

For PrimeKG, each certificate is defined by a reference DAG $\mathcal{G}_{ref} = (\mathcal{N}_{ref}, \mathcal{E}_{ref})$ that encodes a specific clinical reasoning pattern. Following Algorithm 3, we generate isomorphic DAGs $\mathcal{G}'$ that preserve the structure and relation types of $\mathcal{G}_{ref}$ while varying the specific entities.

Formally, for a given $\mathcal{G}_{ref}$, we sample DAGs $\mathcal{G}'$ such that there exists a bijection $\phi : \mathcal{N}_{ref} \to \mathcal{N}'$ preserving: 1. Edge structure: $(v_1, v_2) \in \mathcal{E}_{ref} \iff (\phi(v_1), \phi(v_2)) \in \mathcal{E}'$ 2. Relation types: $(v_1, v_2)_{\mathcal{A}} = (\phi(v_1), \phi(v_2))_{\mathcal{A}}$

We define nine core reference patterns capturing common clinical reasoning chains:

- Which drug is a contraindication for #disease A# and indication for #disease B#, when patient has both diseases?
- Which drug can be used as indication for ¡disease A¿ and is atleast an off-label use drug for #disease B#?
- Which drug can be used as an indication for #disease A# and is an indication for #disease B#?
- Which drug is an indication for #disease A# and interacts synergistically with the treatment of #disease B#?
- Which drug targets #genes/proteins(RIDR33)# associated with #disease(RIDR38)#?
- Which disease is contraindicated by drug indication #drug A# for #disease A#, which both show #effect/phenotype A#?
- Which disease is treated with #drug A# but contraindicated with #drug B#?
- Which drug can be used to treat a disease caused by exposure of #exposure A#?
- Which disease can be a diagnosis for the following phenotypes - #list of phenotypes#?

For each reference DAG $\mathcal{G}_{ref}$, we implement specialized sampling functions that generate isomorphic DAGs while respecting PrimeKG's medical domain constraints. These functions implement the `generateIsomorphisms` procedure from Algorithm 3 as follows:

1. Type constraints: Each node $v \in \mathcal{N}_{ref}$ specifies required entity types (e.g., DRUG, DISEASE) 2. Relation constraints: Each edge $e \in \mathcal{E}_{ref}$ requires specific medical relationships (e.g., indication, contraindication) 3. Clinical validity: Additional constraints ensure sampled DAGs represent valid clinical scenarios

For implementation details of these specialized sampling functions, see the discovery_functions in our codebase.

Query Formulation: For a sampled DAG $\mathcal{G}'$, we generate natural language queries using predefined templates τ (line 4 in Algorithm 1). Each certificate has a minimum of 5 semantically equivalent templates that vary in:

1. Syntactic structure: "Which drug treats X and Y?" vs. "Find a medication indicated for both X and Y" 2. Clinical terminology: "contraindication" vs. "should not be used with" 3. Question framing: Direct vs. scenario-based queries

8.5 Prompt Construction

The final prompt is constructed using a template applied to the query. This process involves several steps, each addressing specific requirements:

- Query Formulation: Convert the generated path into a query string as described earlier.
- Context: This is the supporting text we provide the LLM to answer the query correctly. We additionally trim the context to fit within the LLM's context length limits.
- Few-shot Examples: Include examples to guide the LLM in understanding the query format and expected answer structure.

- Answer Options Generation: Create a set of possible answers, including the correct one. The LLM has to choose one of these options as the correct one.
- Distractors: In the distractor setting, we need to find distractors for the query which need to be included in the prompt.

These inclusions ensure that the prompt is comprehensive, fits within model constraints, and provides sufficient guidance for the LLM to generate accurate responses. We also provide information on the aliases used and the entities they correspond to in the prompt, to ensure that the LLM knows about the alias.

8.5.1 Distractor Selection

Following Definition 2.2, we implement distinct distractor selection strategies for each knowledge graph structure to create challenging evaluation scenarios.

Wikidata5m Linear Paths For a path $\mathcal{N} = [v_1, ..., v_k]$, we select distractors through weighted sampling based on path position:

1. **Distractor Identification**: For each node $v_i \in \mathcal{N}$: - Find candidate distractors $D_i = \{\tilde{v} \mid (v_i, \tilde{v}) \in \mathcal{E} \wedge \tilde{v} \notin \mathcal{N}\}$ - Filter to preserve relation type: $D'_i = \{\tilde{v} \in D_i \mid \exists j > i : (v_i, \tilde{v})_{\mathcal{A}} = (v_i, v_j)_{\mathcal{A}}\}$

2. **Position-based Weighting**: Assign weights to $\tilde{v} \in D'_i$:

$$w(\tilde{v}) = i/|\mathcal{N}|$$

This weights distractors near path end higher, as formalized in Algorithm 6.

PrimeKG DAG Structure For reference DAG $\mathcal{G}_{ref}$ and its isomorphic instance $\mathcal{G}'$, we:

1. Include all valid distractors in context:

$$D = \{\tilde{v} \mid \exists v \in \mathcal{N}' : (\tilde{v} \text{ satisfies Definition 2.2})\}$$

2. Ensure answer options contain at least one distractor:

$$\text{options} = \{\mathcal{N}'_{sink}\} \cup \{d \sim \text{Uniform}(D)\} \cup \text{Others}$$

This flexible approach demonstrates how our framework is easily adaptable to different kinds of queries.

8.5.2 Answer Options

After formulating the query, we generate a set of answer options. This set includes:

- The correct answer: The last entity in the generated path.
- Other entities in the path.
- Related entities: Entities that share some edge with nodes in the path but are not part of the path.
- Distractors: A distractor is a node in the knowledge graph $\mathcal{G}$ that shares a relation with a node in the path, mirroring the relation that continues the path, but the distractor is not itself part of the path. For a formal definition, refer to Definition 2.2. These are only included in the options in the distractor setting.

The process of generating answer options is detailed in Algorithm 5. In the algorithm, we sample answer options from the set described above. The answer option algorithm assumes that distractors are input in a list according to the order of preference. Another important note is that for primeKG, we try to ensure that all options are of the same type as the correct answer to prevent the LLM from easily eliminating wrong options.

Algorithm 5 Generate Answer Options

```
1: Input: $correct\_ans$, $distractors$, $path\_entities$, $random\_entities$, $Graph$, $min\_num\_options$
2: Output: $options$
3: $options \leftarrow [(correct\_ans] \cup distractors$
4: Add path entities to $options$
5: Add random entities from $random\_entities$ to $options$
6: Return Shuffle($options$[: $min\_num\_options$])
```

Algorithm 6 Get Best Distractor

```
1: Input: Graph $G$, Path $\mathcal{N}$
2: Output: $best\_distractor$
3: $D \leftarrow []$ {List of distractors}
4: $W \leftarrow []$ {Weights for distractors}
5: for $i \leftarrow 0$ to len($\mathcal{N}$) $-2$ do
6:     $v \leftarrow \mathcal{N}[i]$
7:     $N \leftarrow$ GetNeighbors($G$, $v$)
8:     $N\_distractors \leftarrow$ FilterDistractors($N$, $v$, $\mathcal{N}$)
9:     Extend $D$ with $N\_distractors$
10:    Extend $W$ with $[i+1] * len(N\_distractors)$
11: end for
12: if $D$ is not empty then
13:     Return WeightedRandomChoice($D$, $W$)
14: else
15:     Return None
16: end if
```

8.6 Context Trimming

To address the input length limitations of various LLMs, we implement a context trimming procedure. Including all text associated with each node in a reasoning path can result in excessively long contexts. Our procedure aims to preserve the most relevant information from the knowledge graph and supporting texts while respecting each model's maximum input length. This involves identifying relevant sentences per edge in the graph and then trimming the context for each query based on this information.

8.6.1 Finding Relevant Sentences per Edge (Wikidata5m)

Each node in the Wikidata5m knowledge graph has associated textual support for its relations. We utilize this textual information to provide query-relevant context. We need to determine the relevant information from the textual supports for each edge as this would help us trim the contexts accordingly. For each edge (u, v) in the knowledge graph used in the query or answer options generation, we perform the following steps:

1. **Collect Aliases and Text:** We gather aliases and the associated text paragraphs for both nodes u and v.

2. **Split into Sentences:** We split the text paragraphs of u and v into individual sentences using NLTK.

3. **Identify Relevant Sentences:** We identify sentences that explicitly link the two nodes. A sentence from u's text is considered relevant if it contains an alias of v, and vice versa.

4. **Discard Edges without Relevant Sentences:** If no relevant sentences are found for an edge, it is deemed unsupported and is discarded from the graph.

5. **Prepend First Sentence:** To ensure the entity's primary name or common alias is included, we prepend the first sentence of each node's text to its list of relevant sentences.

8.6.2 Relevant sentences per Edge PrimeKG

As we synthetically generate data for each edge in PrimeKG, we already have a sentence that corresponds to an edge in primeKG. So no further processing is needed.

8.6.3 Trimming to Fit Context Length

When constructing the final prompt for the LLM, we prioritize including the most relevant information within the model's context length limit. Therefore we need to trim the context according to the LLM's context limit. We use the following procedure (detailed in Algorithm 8.6.3):

1. **Create Sentence Lists:** We create three lists of sentences:
   - S_{all}: Contains all sentences from the text paragraphs of nodes involved in the query and answer options.
   - S_{query}: Contains all relevant sentences for the edges that constitute the query path.
   - $S_{options}$: Contains all relevant sentences for the edges used to generate the answer options.
2. **Construct the Final Context:**
   (a) We prioritize including all sentences from S_{query} as they are directly related to the query.
   (b) Next, we add as many sentences from $S_{options}$ as possible, given the remaining context length limit.
   (c) Finally, we fill the remaining space with sentences from S_{all} that have not been included yet, ensuring no sentence is repeated.

Algorithm 7 Context Construction

```
1: Input: $S_{all}$, $S_{query}$, $S_{options}$, $L_{max}$
2: Output: $C_{trimmed}$
3: $C \leftarrow S_{query}$
4: ASSERT TokenizedLength($C$) $\leq L_{max}$
5: $S_{seen} \leftarrow$ UniqueSet($C$)
6: for each $s$ in $S_{option}$+$S_{all}$ do
7:     if $s \notin S_{seen}$ and TokenizedLength($C + s$) $\leq L_{max}$ then
8:         $C \leftarrow C + s$
9:         Add $s$ to $S_{seen}$
10:    end if
11: end for
12: return $C$ as $C_{trimmed}$
```

Vanilla Setting Note: In the vanilla setting, we only want to check simple knowledge comprehension abilities of LLMs, so if our DAG used to create a query has an edge type (alias), we ensure that the provided context trims any sentences which are in the relevant sentences of the same edge type has any edge type in the DAG. In **distractor** setting, there is no such constraint.

8.7 Few-Shot Examples

To guide the LLM towards the desired response format and demonstrate the reasoning process, we include 2 few-shot examples in the prompt (common across all prompts to all models). These examples provide a clear illustration of how to approach the knowledge comprehension task. We investigate the impact of varying number of few-shot examples on LLM performance in Appendix 9. We use the following few-shot examples for wikidata5m:

Few Shot Examples

Common Context: entity_B is the son of entity_A. entity_E is the sister of entity_A. entity_B leads entity_C. Entity_D is a member of Entity_C. Entity_D is a friend of entity_E. entity_E has mother entity_F who likes the services of entity_C.

Question 1: entity_A → `(father of)` → `(leader of)` → ?
Options: 1. entity_F, 2. entity_C, 3. entity_D, 4. entity_E, 5. entity_B
Answer: 2. **entity_C**
Explanation: entity_A → `(father of)` entity_B → `(leader of)` entity_C
How to get answer: Find who entity_A is father of to get entity_B, then find what B is the leader of to get entity_C.

Question 2: entity_B → `(chief of)` → `(constitutes)` → `(companion of)` → ?
Options: 1. entity_F, 2. entity_C, 3. entity_D, 4. entity_E, 5. entity_A
Answer: 4. **entity_E**
Explanation: entity_B → `(chief of)` entity_C → `(constitutes)` entity_D → `(companion of)` entity_E
How to get answer: Find what entity_B is the chief of to get entity_C, find what entity_C constitutes to get entity_D, then find the companion of entity_D to get entity_E.

We use the following few-shot examples for primeKG:

Few Shot Examples

Common Context: (drug) entity_A has side effects: effect_1, effect_5. (disease) entity_C has contraindication relation to (drug) entity_E. (disease) entity_C is indicated for (drug) entity_A. (drug) entity_D has side effects: effect_2, effect_3, effect_4. (drug) entity_D has indication for (disease) entity_C. (drug) entity_E has indication to (disease) entity_B. (gene/protein) entity_F is carrier for (drug) entity_E. (disease) entity_B is indicated for (drug) entity_G.

Question 1: Which drug → `(has indication for)` (disease) entity_C and has → `(side effect)` effect_5 ?
Options: 1. entity_F,
n 2. entity_E,
n 3. entity_D,
n 4. entity_A,
n 5. entity_B
Answer: 2. **entity_A**
Explanation: entity_A has the side effects effect_5 while it is the indication of entity_C, so answer is entity_A.
How to get answer: Find the drugs entity_C is indicated for, which gives us entity_A and then check which of them has the side effect_5.

Question 2: What drugs should be avoided in (disease) entity_C but are indicated for (disease) entity_B?
Options: 1. entity_F,
n 2. entity_A,
n 3. entity_D,
n 4. entity_E,
n 5. entity_B
Answer: 4. **entity_E**
Explanation: entity_B is indicated for entity_E (drug), and entity_E has contraindication to entity_C.
How to get answer: find the drugs entity_B is indicated for to get entity_E, and entity_G. then find the contraindications of entity_C to get entity_E. The intersection of these two sets gives the answer, which in this case is entity_E.

8.8 Final Prompt

The final prompt presented to the LLM is constructed using a template (same for wikidata5m and primeKG) that incorporates several key elements:

Trimmed Context [8.6]: The relevant context extracted and trimmed.

Query [8.4]: The multi-hop query.

Answer Options [8.5.2]: The generated answer options, including the correct answer and distractors.

Few-Shot Examples [8.7]: A set of examples demonstrating the desired response format.

The prompt template is structured as follows:

Prompt Template

{few_shot_examples}
Actual Query: Given Context: {context}
Answer the question: {query}
answer the question by selecting the correct answer from the following options:
{options}

The format for beginning your response is:
correct answer: $<$ *option_number* $>$. $<$ *answer* $>$, *because* $<$ *succinct reason* $>$
follow this exact format and only choose from the given options

Estimating the number of unique prompts: We estimate a lower bound on the number of unique prompts that can be generated from the Wikidata5m Knowledge Graph (KG) by quantifying the potential unique queries within the graph. Each query can be formulated into multiple prompts through variations in answer presentation, thus making query count a conservative estimate. We analyzed the 50 subgraphs employed in our experiments. For each subgraph, we calculated the number of unique paths(upto the maximum path length hyperparameter, $\rho = 4$) and calculated the number of possible queries for each path using the number of aliases for each each entity and relation within a path. This analysis provides an estimate of the unique query generation capacity inherent in subgraphs in our KG.

The mean number of unique queries was 3.04×10^{15} with a median of 1.24×10^{15}. The minimum and maximum observed values were 1.36×10^{12} and 1.46×10^{16}, respectively.

Importantly, these figures conservatively estimate the number of unique prompts, as they only consider query variations and not the diversity introduced by different answer options. The actual number of unique prompts is likely significantly larger, making exhaustive enumeration of all possible generated prompts infeasible.

8.9 Response Checker function

We implement a simple response checker function to evaluate the correctness of the model's answers. The function is defined in Algorithm 8.9. We write a regular expression to account for trivial formatting errors like extra spaces, brackets, incorrect punctuation, etc.

Algorithm 8 Response Checker

```
1: Input: $model\_answer$, $correct\_answer\_num$
2: Output: $is\_correct$
3: $model\_answer \leftarrow$ LowerCase($model\_answer$)
4: $correct\_answer\_num \leftarrow$ LowerCase(ToString($correct\_answer\_num$))
5: $pattern \leftarrow$ SpecializedRegularExpression(”correct answer: ” + $correct\_answer\_num$)
6: if RegexMatch($pattern$, $model\_answer$) then
7:     $is\_correct \leftarrow 1$
8: else
9:     $is\_correct \leftarrow 0$
10: end if
11: return $is\_correct$
```

9 Ablations

9.1 Few Shot Prompts

We conduct an ablation study to examine the impact of varying the number of few-shot examples on Gemini-Flash's performance in the vanilla task setting. While our default configuration uses two few-shot examples, we extend this analysis to include up to five examples. Interestingly, we observe no significant variation in performance across these different few-shot configurations. The results are presented below in 2.

9.2 Distractor Distributions

To assess the impact of distractor distribution on model performance, we implement three distinct distractor distribution strategies:

1. Tail-weighted: Linearly increasing weights towards the tail end of the path, prioritizing distractors near the answer node. This serves as our default setting.
2. Head-weighted: Linearly increasing weights towards the head of the path, emphasizing distractors near the query's starting point.
3. Uniform: Equal probability of selecting distractors from any position along the path.

We observe no significant differences in either of the settings. The results are presented in 3 below.

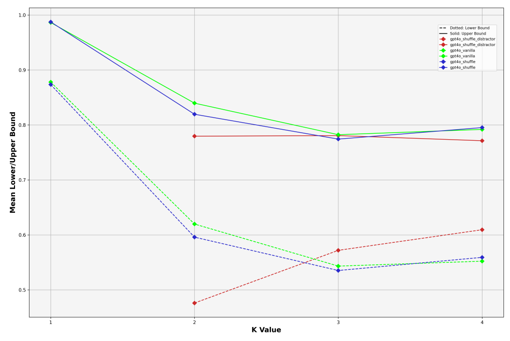


Figure 7: Variations in the bounds against the path lengths across various specifications.

9.3 Model Performances with varying Path Length

Among our certificates, we have queries of various lengths. Here we study the effects on models behavior on queries with varying length by considering the number of hops they require to reason to answer the query(which is 1 less than the path length). To do so, we refer to the number of hops to answer a query as k where $1 \leq k < \rho$.

Varying Setting: In figure 7 we show plots for various specifications for the GPT4o model.

Varying Quantization: In figure 8 we show plots when the quantization is varied with the Llama3-8B model on the shuffle specification and their effects on performance.

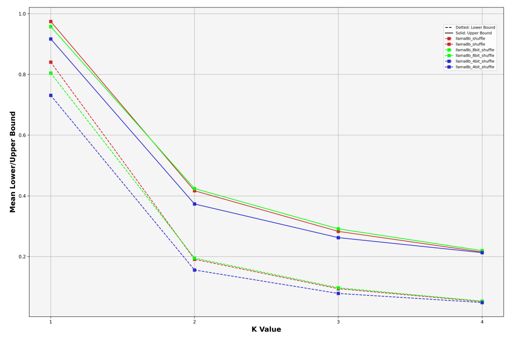


Figure 8: Variations in the bounds against the path lengths across various quantizations.

Varying Models: In fig 9 we show plots for the shuffle specification and performance across the models(the open-source models use fp16 precision).

Table 2: Certification results for LLMs in vanilla setting with different number of few-shot examples

Model	Avg. lower bound	Avg. upper bound	Avg. accuracy
Gemini-1.5-Flash 2Shot (Default)	0.46 ± 0.06	0.58 ± 0.06	0.52 ± 0.06
Gemini-1.5-Flash 3Shot	0.46 ± 0.06	0.58 ± 0.06	0.52 ± 0.06
Gemini-1.5-Flash 4Shot	0.46 ± 0.07	0.58 ± 0.07	0.52 ± 0.07
Gemini-1.5-Flash 5Shot	0.46 ± 0.07	0.58 ± 0.07	0.52 ± 0.07

Table 3: Certification results for Gemini-Flash with different distractor distributions

Model	Avg. lower bound	Avg. upper bound	Avg. accuracy
Gemini-1.5-Flash Setting 1 (Default)	0.42 ± 0.10	0.55 ± 0.10	0.48 ± 0.10
Gemini-1.5-Flash Setting 2	0.42 ± 0.11	0.55 ± 0.11	0.48 ± 0.11
Gemini-1.5-Flash Setting 3	0.42 ± 0.11	0.55 ± 0.11	0.48 ± 0.11

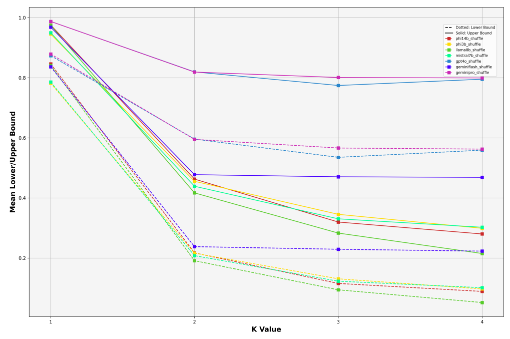


Figure 9: Variations in the bounds against the path lengths across various models in the shuffle setting.

In Figure 7, we observe that the performance across settings converges as k increases and the distractor setting is most impactful on the performance for $k = 2$.

In Figure 8, we infer that as k increases the performance of the models' on the task converges across the different quantizations. We hypothesize this is due to the increasing complexity of the reasoning task.

In Figure 9, we see that larger models (GPT-4o, Gemini-Pro) show less severe drop in performance compared to their smaller models. The figure shows that large models may have learnt to better apply 1-step reasoning for multiple steps when compared to their smaller counterparts.

9.4 Chain of Thought Prompting

We also conduct an ablation on how Chain-of-Thought(COT) prompting can affect the performance of language models on the knowledge comprehension task. Specifically, we investigate the Phi-3 3B model (precision: float16) in the vanilla setting with COT prompting strategy. We augmented our standard few-shot examples (8.7) with COT steps and added structured reasoning guidance to the prompt template (8.5):

COT additions to prompt template

Answer in the following the below format:
Let's solve this step by step: 1) Let's identify the starting point and path: - Start: [identify starting entity] - Path to follow: [break down the path components]
2) Let's follow the path: Starting from [entity] $\rightarrow$ [first relationship] $\rightarrow$ [next entity] $\rightarrow$ [next relationship] $\rightarrow$ [next entity] ... [continue as needed]
3) Verify our final destination reaches one of the given options
Therefore, the correct answer is: <option_number>. <option_text>

In the vanilla setting, adding COT prompting improved Phi-3 3B's performance, with the bounds increasing by 0.11 summarized in Table 4. While we acknowledge the potential benefits of COT, earlier experiments were limited due to the significantly increased computational cost (generating 5-8 times more tokens) and the expenses of COT, particularly with closed-source models as output tokens are much more expensive.

Table 4: Certification results for Phi-3 3B with and without COT

Prompting Strategy	Avg. lower bound	Avg. upper bound	Avg
No COT (Default)	0.34 ± 0.05	0.46 ± 0.06	
COT	0.45 ± 0.08	0.57 ± 0.08	

9.5 Choice of 250 Samples for Certification

In this section we show how increasing sample size beyond 250 provides diminishing results in certification stability. We show this variation for the Gemini-Flash model on the Vanilla specifications in the table below.

Table 5: Variation for certification results with increasing samples

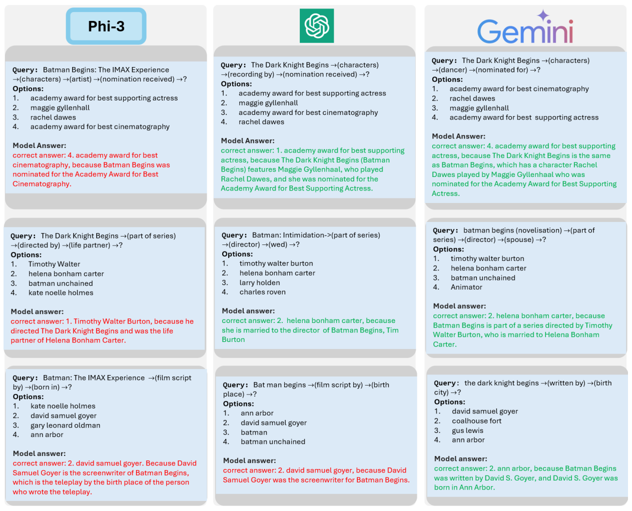
Num Questions	Low Bound Std	Upper Bound Std
50	0.16	0.15
100	0.11	0.10
200	0.07	0.07
250	0.06	0.06

10 Case Studies from certifying over Wikidata5m

We analyze the certification results, qualitatively. First, we show the responses of 3 models in Figure 10 — Phi-3 (3B), GPT-4o, and Gemini, obtained when certifying them for the Vanilla specification defined over a subgraph pivoted at the node for 'Batman Begins' movie. The samples reflect the certificates. Next, we identify and categorize prominent kinds of model responses. We frequently see the following failure modes — *distracted* and *missed relation*. In the former, the model gets deviated from the query by following the distractor context in its prompt, resulting in an incorrect answer. In the latter, the model skips some reasoning steps needed for the final correct answer. In cases of *good reasoning*, model accurately follows the query and gives the correct answer. Figure 11 presents examples of the aforementioned kinds of model responses for GPT-4o.

11 Additional Case Studies from certifying over PrimeKG

We show the responses of 3 models in Figure 12 — Phi-3 (3B), GPT-4o, and Gemini-Pro, obtained when certifying them for the Distractor specification. We see occurrences of various models having different robustness against distractor information.



Phi-3

Query: Batman Begins: The IMAX Experience →(characters) →(artist) →(nomination received) →?
Options:
1. academy award for best supporting actress
2. maggie gyllenhall
3. rachel dawes
4. academy award for best cinematography

Model Answer:
correct answer: 4. academy award for best cinematography, because Batman Begins was nominated for the Academy Award for Best Cinematography.

Query: The Dark Knight Begins →(part of series) →(directed by) →(life partner) →?
Options:
1. Timothy Walter
2. helena bonham carter
3. batman unchained
4. kate noelle holmes

Model answer:
correct answer: 1. Timothy Walter Burton, because he directed The Dark Knight Begins and was the life partner of Helena Bonham Carter.

Query: Batman: The IMAX Experience →(film script by) →(born in) →?
Options:
1. kate noelle holmes
2. david samuel goyer
3. gary leonard oldman
4. ann arbor

Model answer:
correct answer: 2. david samuel goyer. Because David Samuel Goyer is the screenwriter of Batman Begins, which is the teleplay by the birth place of the person who wrote the teleplay.

Query: The Dark Knight Begins →(characters) →(recording by) →(nomination received) →?
Options:
1. academy award for best supporting actress
2. maggie gyllenhall
3. academy award for best cinematography
4. rachel dawes

Model Answer:
correct answer: 1. academy award for best supporting actress, because The Dark Knight Begins (Batman Begins) features Maggie Gyllenhaal, who played Rachel Dawes, and she was nominated for the Academy Award for Best Supporting Actress.

Query: Batman: Intimidation->(part of series) →(director) →(wed) →?
Options:
1. timothy walter burton
2. helena bonham carter
3. larry holden
4. charles roven

Model answer:
correct answer: 2. helena bonham carter, because she is married to the director of Batman Begins, Tim Burton

Query: Bat man begins →(film script by) →(birth place) →?
Options:
1. ann arbor
2. david samuel goyer
3. batman
4. batman unchained

Model answer:
correct answer: 2. david samuel goyer, because David Samuel Goyer was the screenwriter for Batman Begins.

Gemini

Query: The Dark Knight Begins →(characters) →(dancer) →(nominated for) →?
Options:
1. academy award for best cinematography
2. rachel dawes
3. maggie gyllenhall
4. academy award for best supporting actress

Model Answer:
correct answer: 4. academy award for best supporting actress, because The Dark Knight Begins is the same as Batman Begins, which has a character Rachel Dawes played by Maggie Gyllenhaal who was nominated for the Academy Award for Best Supporting Actress.

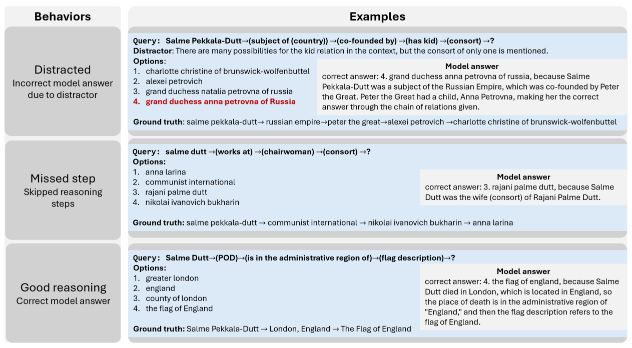
Query: batman begins (novelisation) →(part of series) →(director) →(spouse) →?
Options:
1. timothy walter burton
2. helena bonham carter
3. batman unchained
4. Animator

Model answer:
correct answer: 2. helena bonham carter, because Batman Begins is part of a series directed by Timothy Walter Burton, who is married to Helena Bonham Carter.

Query: the dark knight begins →(written by) →(birth city) →?
Options:
1. david samuel goyer
2. coalhouse fort
3. gus lewis
4. ann arbor

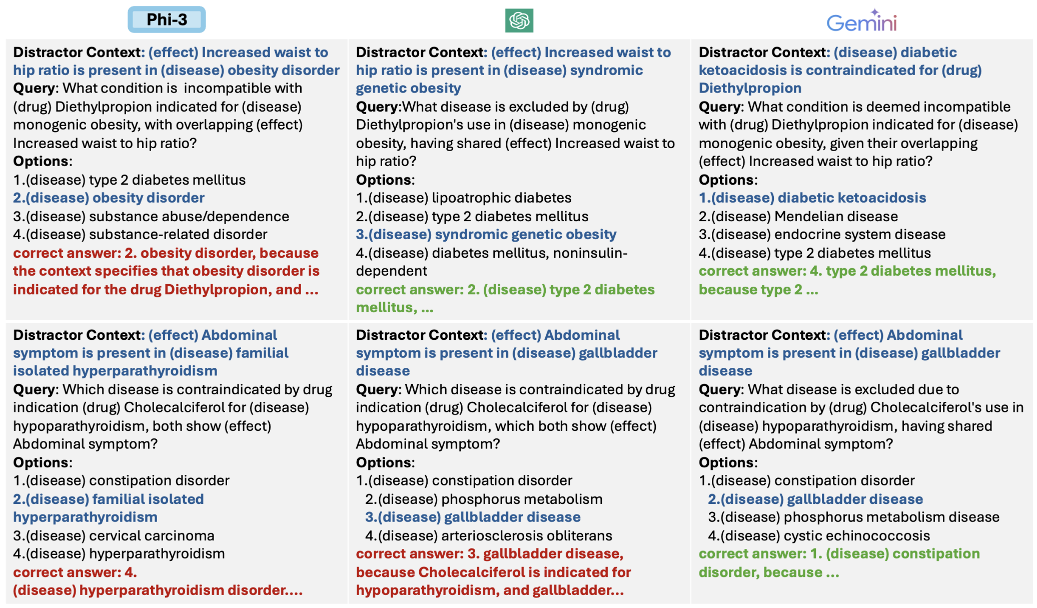
Model answer:
correct answer: 2. ann arbor, because Batman Begins was written by David S. Goyer, and David S. Goyer was born in Ann Arbor.

Figure 10: Qualitative analysis of samples used for certifying knowledge comprehension for Vanilla specifications on the Wikidata5m subgraph pivoted at the node for 'Batman Begins' movie. The context provided in the prompts is not shown for brevity. Wrong model responses are colored red and correct ones are colored green. The samples are consistent with our results, wherein Phi-3 (3B) has lower certification bounds than GPT-4o's bounds, which are lower than those for Gemini-Pro.



Behaviors	Examples
Distracted Incorrect model answer due to distractor	**Query: Salme Pekkala-Dutt→(subject of (country)) →(co-founded by) →(has kid) →(consort) →?** **Distractor**: There are many possibilities for the kid relation in the context, but the consort of only one is mentioned. **Options**: 1. charlotte christine of brunswick-wolfenbuttel 2. alexei petrovich 3. grand duchess natalia petrovna of russia **4. grand duchess anna petrovna of Russia** **Model answer** correct answer: 4. grand duchess anna petrovna of russia, because Salme Pekkala-Dutt was a subject of the Russian Empire, which was co-founded by Peter the Great. Peter the Great had a child, Anna Petrovna, making her the correct answer through the chain of relations given. **Ground truth**: salme pekkala-dutt→ russian empire→peter the great→alexei petrovich →charlotte christine of brunswick-wolfenbuttel
Missed step Skipped reasoning steps	**Query: salme dutt →(works at) →(chairwoman) →(consort) →?** **Options:** 1. anna larina 2. communist international 3. rajani palme dutt 4. nikolai ivanovich bukharin **Model answer** correct answer: 3. rajani palme dutt, because Salme Dutt was the wife (consort) of Rajani Palme Dutt. **Ground truth:** salme pekkala-dutt → communist international → nikolai ivanovich bukharin → anna larina
Good reasoning Correct model answer	**Query: Salme Dutt→(POD)→(is in the administrative region of)→(flag description)→?** **Options:** 1. greater london 2. england 3. county of london 4. the flag of England **Model answer** correct answer: 4. the flag of england, because Salme Dutt died in London, which is located in England, so the place of death is in the administrative region of "England," and then the flag description refers to the flag of England. **Ground truth:** Salme Pekkala-Dutt → London, England → The Flag of England

Figure 11: GPT-4o responses showing good reasoning, failures due to distractors or skipped reasoning



Phi-3		Gemini
Distractor Context: **(effect) Increased waist to hip ratio is present in (disease) obesity disorder** **Query**: What condition is incompatible with (drug) Diethylpropion indicated for (disease) monogenic obesity, with overlapping (effect) Increased waist to hip ratio? **Options**: 1.(disease) type 2 diabetes mellitus **2.(disease) obesity disorder** 3.(disease) substance abuse/dependence 4.(disease) substance-related disorder **correct answer: 2. obesity disorder, because the context specifies that obesity disorder is indicated for the drug Diethylpropion, and ...**	**Distractor Context**: **(effect) Increased waist to hip ratio is present in (disease) syndromic genetic obesity** **Query**:What disease is excluded by (drug) Diethylpropion's use in (disease) monogenic obesity, having shared (effect) Increased waist to hip ratio? **Options**: 1.(disease) lipoatrophic diabetes 2.(disease) type 2 diabetes mellitus **3.(disease) syndromic genetic obesity** 4.(disease) diabetes mellitus, noninsulin-dependent **correct answer: 2. (disease) type 2 diabetes mellitus, ...**	**Distractor Context**: **(disease) diabetic ketoacidosis is contraindicated for (drug) Diethylpropion** **Query**: What condition is deemed incompatible with (drug) Diethylpropion indicated for (disease) monogenic obesity, given their overlapping (effect) Increased waist to hip ratio? **Options**: **1.(disease) diabetic ketoacidosis** 2.(disease) Mendelian disease 3.(disease) endocrine system disease 4.(disease) type 2 diabetes mellitus **correct answer: 4. type 2 diabetes mellitus, because type 2 ...**
Distractor Context: **(effect) Abdominal symptom is present in (disease) familial isolated hyperparathyroidism** **Query**: Which disease is contraindicated by drug indication (drug) Cholecalciferol for (disease) hypoparathyroidism, both show (effect) Abdominal symptom? **Options**: 1.(disease) constipation disorder **2.(disease) familial isolated hyperparathyroidism** 3.(disease) cervical carcinoma 4.(disease) hyperparathyroidism **correct answer: 4. (disease) hyperparathyroidism disorder....**	**Distractor Context**: **(effect) Abdominal symptom is present in (disease) gallbladder disease** **Query**: Which disease is contraindicated by drug indication (drug) Cholecalciferol for (disease) hypoparathyroidism, which both show (effect) Abdominal symptom? **Options**: 1.(disease) constipation disorder 2.(disease) phosphorus metabolism **3.(disease) gallbladder disease** 4.(disease) arteriosclerosis obliterans **correct answer: 3. gallbladder disease, because Cholecalciferol is indicated for hypoparathyroidism, and gallbladder...**	**Distractor Context**: **(effect) Abdominal symptom is present in (disease) gallbladder disease** **Query**: What disease is excluded due to contraindication by (drug) Cholecalciferol's use in (disease) hypoparathyroidism, having shared (effect) Abdominal symptom? **Options**: 1.(disease) constipation disorder **2.(disease) gallbladder disease** 3.(disease) phosphorus metabolism disease 4.(disease) cystic echinococcosis **correct answer: 1. (disease) constipation disorder, because ...**

Figure 12: Samples from PrimeKG's Distractor specifications. Full context is omitted for brevity; only distractor texts are shown. The correct answer is in blue, with wrong/correct responses in red/green. Results match certificates, where Phi-3 (14B) yields lower bounds than GPT-4o and Gemini-Pro.