Verifiable, Debuggable, and Repairable Commonsense Logical Reasoning via LLM-based Theory Resolution

Anonymous ACL submission

Abstract

Recent advances in Large Language Models (LLM) have led to substantial interest in their application to commonsense reasoning tasks. Despite their potential, LLMs are susceptible to reasoning errors and hallucinations that may be harmful in use cases where accurate reasoning is critical. This challenge underscores the need for verifiable, debuggable, and repairable LLM reasoning. Recent works have made progress toward verifiable reasoning with LLMs by using them as either (i) a reasoner over an axiomatic knowledge base, or (ii) a semantic parser for use in existing logical inference systems. However, both settings are unable to extract commonsense axioms from the LLM that are not already formalized in the knowledge base, and also lack a reliable method to repair missed commonsense inferences. In this work, we present LLM-TRes, a logical reasoning framework based on the notion of "theory resolution" that allows for seamless integration of the commonsense knowledge from LLMs with a verifiable logical reasoning framework that mitigates hallucinations and facilitates debugging of the reasoning procedure as well as repair. We crucially prove that repaired axioms are theoretically guaranteed to be given precedence over flawed ones in our theory resolution inference process. We conclude by evaluating on three diverse language-based reasoning tasks – preference reasoning, deductive reasoning, and causal commonsense reasoning – and demonstrate the superior performance of LLM-TRes vs. state-of-the-art LLM-based reasoning methods in terms of both accuracy and reasoning correctness.

1 Introduction

The rise of Large Language Models (LLMs) has marked a pivotal moment in the real-world deployment of AI, particularly due to the exceptional ability of LLMs to handle complex reasoning tasks (Chang et al., 2024; Huang and Chang, 2023). Research has shown that LLMs have acquired significant commonsense knowledge (Zhao et al., 2024; Bian et al., 2023), which is crucial for engaging with real-world users in tasks such as question answering (Singhal et al., 2023) and recommendation (Sanner et al., 2023). Unfortunately, LLMs are prone to a variety of reasoning errors; for example, they commonly incorporate superficially plausible but factually incorrect information into their reasoning in a phenomenon known as *hallucination* (Zhang et al., 2023b; Ji et al., 2023). Furthermore, since the underlying reasoning process of the LLM is latent and hence largely opaque, validating reasoning soundness and identifying errors remains an open research problem. Such issues present a significant challenge to the reliability of using LLMs as reasoning systems, which impedes their practical utility (Mallen et al., 2023).

In light of these obstacles, recent research has proposed methodologies for extracting verifiable reasoning from LLMs by leveraging formal reasoning procedures. Such works fall under two main categories: (i) Using the LLM as a reasoner across an axiomatic knowledge base, while organizing the reasoning process into simpler subgoals to facilitate soundness of the overall reasoning (Kazemi et al., 2023). (ii) Leveraging the LLM as a semantic parser that translates natural language statements into logical axioms, followed by the use of an off-the-shelf theorem prover to perform logical reasoning (Pan et al., 2023; Olausson et al., 2023). While these seminal works have made progress towards verifiable LLM reasoning, their application in real-world tasks requiring commonsense reasoning is limited since they all suffer from the inability to extract verifiable commonsense axioms from the LLM that are not already formalized in the provided knowledge base axioms. Hence, these existing methodologies critically lack the ability to leverage the LLM as a verifiable commonsense reasoner to fill-in inevitable knowledge base gaps.

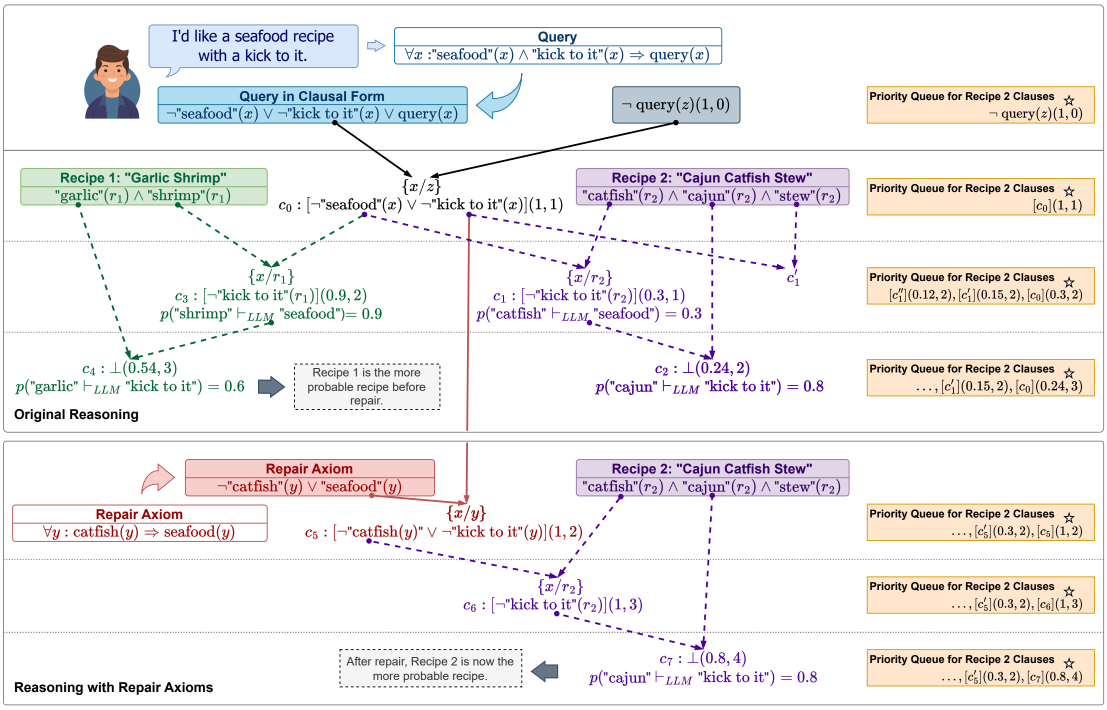

Figure 1: Preference reasoning is used as an illustrative example to show the LLM-TRes workflow. Top: LLM-based theory resolution is performed to calculate proof scores of two candidate Recipes entailing the Query. The proof begins from the negated query, and for each resolvent clause, a priority score tuple: (*proof plausibility score, proof length*) is calculated and pushed to a priority queue (only Recipe 2 clauses are shown here). At each step, the clause with the highest priority in the queue becomes the active clause. Here, due to a flawed low probability assigned to *"catfish"* entailing *"seafood"*, the *proof score* of Recipe 2 is mistakenly calculated lower than it should be. Bottom: After insertion of the Repair Axiom, the erroneous reasoning is repaired, leading to a higher score for the correct Recipe 2.

Furthermore, these methods lack any mechanism for repairing reasoning mistakes after detection.

To address these challenges, we propose LLM-TRes, a formal reasoning framework using LLMs. LLM-TRes satisfies three key desiderata that we illustrate through the worked example in Figure 1: (i) *Verifiability*: allowing for verification of every step in the reasoning process (i.e., from each successful refutation resolution $\perp$, one can backtrace the entire proof of the Query). (ii) *Debuggability*: being able to identify the incorrect inferences that led to a reasoning mistake (i.e., we observe an incorrect low probability LLM inference that "catfish" entails "seafood" for Recipe 2). (iii) *Repairability*: enabling a deterministic and reliable mechanism for rectifying the identified errors to produce correct inference (e.g., once we add the the explicit Repair Axiom $\forall y$ "catfish"$(y) \Rightarrow$ "seafood"(y), we arrive at a much higher proof plausibility for the correct preference match of the query to Recipe 2).

Formally, LLM-TRes is based on the concept of *theory resolution* (Stickel, 1985; Baumgartner, 1992), drawn from classical logical reasoning that enables the integration of specialized reasoners into the resolution theorem-proving inference rule. Leveraging theory resolution, LLM-TRes seamlessly incorporates LLMs as specialized reasoners equipped with commonsense knowledge into verifiable logical reasoning. This integration enables extraction of relevant commonsense axioms from the LLM that cannot otherwise be obtained from the knowledge base. Finally, by capitalizing on a specially defined selection rule in our resolution framework, we formally prove that repairing flawed reasoning by the LLM is possible by providing correct axioms that are theoretically guaranteed to override the LLM's flawed reasoning.

In summary, we contribute the following:

- We propose LLM-TRes, a formal reasoning framework founded on theory resolution that allows for incorporating the internal knowledge of the LLMs in a formal reasoning process to mitigate their hallucinations.
- We demonstrate that LLM-TRes provides a fully verifiable and debuggable reasoning

scheme by granting access to all reasoning steps at an atomic level.

- We provide a mechanism for correcting errors in the reasoning process with a theoretical guarantee of prioritizing correct Repair Axioms over incorrect LLM inferences.
- We experiment with LLM-TRes on three distinct tasks – preference reasoning, deductive reasoning, and causal commonsense reasoning – demonstrating superior accuracy and reasoning correctness compared to Chain of Thought (CoT) (Wei et al., 2022) prompting in LLMs (much larger in size) and LAMBADA (Kazemi et al., 2023), a state-of-the-art formal reasoning framework.

2 Related Works

Reasoning with LLMs While their primary design was for text generation, LLMs exhibit remarkable performance in many other NLP tasks that require a variety of reasoning skills (Chang et al., 2024; Xu et al., 2023). Despite such impressive capabilities, errors and hallucinations that can commonly occur in LLM reasoning have motivated research on obtaining dependable reasoning from LLMs while leveraging their intrinsic knowledge. In this regard, several approaches have been proposed to elicit stronger reasoning performance from LLMs such as Chain-of-Thought prompting (Wei et al., 2022; Kojima et al., 2022), Self-Consistency (Wang et al., 2022), Least-to-Most prompting (Zhou et al., 2022), and Selection-Inference (Creswell et al., 2022). Despite being effective in improving reasoning performance, all these methods follow an *informal* reasoning procedure in which the LLM is in charge of performing reasoning and thus does not guarantee the faithfulness of the reasoning process (Shanahan, 2024; Pan et al., 2023). For instance, the reasoning ability of these methods may be unreliable for tasks requiring out-of-domain reasoning (Saparov et al., 2024; Liang et al., 2022), tasks involving negation (Anil et al., 2022), and often degrades with an increase in the length of reasoning steps (Dziri et al., 2024).

Formal Reasoning with LLMs To obtain reliable and verifiable reasoning from LLMs, a number of works have proposed the idea of using LLMs in a *formal* reasoning framework — a systematic and logical process governed by a set of rules and principles (Galotti, 1989). Two main approaches have been proposed in this regard. In the first approach, the LLM is utilized to perform different sub-tasks of a formal logical inference rule to reason over an axiomatic knowledge base. For example, LAMBADA (Kazemi et al., 2023) uses the LLM to perform goal decomposition, rule selection, and fact-checking in a backward chaining process. In a related vein, SymBa (Lee and Hwang, 2024) introduces a top-down solver to control the proof process and uses the LLM as an aide to the solver. In the second approach, LLMs are used as a semantic parser to translate natural language axioms and facts to a specific logical format; here the responsibility of inference is delegated to a symbolic theorem prover. LogicLM (Pan et al., 2023) uses this idea with a self-refinement mechanism to allow the LLM to refine its symbolic conversions. Since LLMs commonly make syntactic and semantic errors in the parsing process, LINC (Olausson et al., 2023) performs majority voting over multiple solutions to obtain the final result.

These works have made significant progress in increasing the reliability and verifiability of LLM reasoning. However, they only utilize axioms that are explicitly provided in the knowledge base, and lack the ability to leverage the intrinsic commonsense knowledge of the LLM by extracting commonsense axioms. This prevents existing methods from incorporating verifiable LLM-derived commonsense knowledge in their reasoning, which is often critical in practical deployed usage. Moreover, these existing methods do not support a reliable mechanism for rectifying incorrect reasoning steps. *We aim to address all of these limitations with our contribution of the LLM-TRes framework.*

3 Methodology: LLM-based Theory Resolution (LLM-TRes)

We first introduce the resolution rule and the concept of theory resolution and then explain our LLM-based Theory Resolution (LLM-TRes) methodology. For the logical knowledge representation in this work, we assume a function- and equality-free first-order logical (FOL) syntax (Chang and Lee, 2014) with all FOL sentences translated to clausal normal form as demonstrated in Figure 1.

Resolution Rule Resolution is a sound inference rule that performs inference by deriving a resolvent clause from two premise clauses containing complementary literals. Given two FOL sentences in clausal form, a new clause can be derived via resolution of their complementary literals, e.g.,

$$\frac{A(x) \vee B(x) \quad \neg B(y) \vee C(y)}{A(x) \vee C(x)}, \tag{1}$$

under the unification $\theta = \{x/y\}$. Following this procedure, new clauses are derived until either a contradiction $\perp$ is found (e.g., deriving both clauses $A(x)$ and $\neg A(x)$ that resolve to $\perp$), or no further resolutions are possible. Finding a contradiction implies that the original set of clauses is inconsistent. Therefore, given the knowledge base $\mathcal{K}$ and a query q, to prove that $\mathcal{K} \vdash q$, one can apply the resolution inference rule to show that $\mathcal{K} \wedge \neg q$ leads to a contradiction $\perp$.

Theory Resolution Theory resolution (Stickel, 1985) is a methodology that enables the integration of special purpose reasoning theories into resolution theorem proving. Based on theory resolution, given two clauses $c_1 = A(x) \vee B(x)$ and $c_2 = C(x) \vee D(x)$, if a theorem prover T identifies $B(x)$ and $\neg C(y)$ under unification $\theta = \{x/y\}$ to be unsatisfiable (i.e., $\forall x B(x) \wedge \neg C(x) \vdash_T \perp$), despite lacking complementary literals with identical predicates, the two clauses can still be resolved:

$$\frac{A(x) \vee B(x) \quad \neg C(y) \vee D(y)}{A(x) \vee D(x)}. \tag{2}$$

Theory resolution considerably broadens the applicability of the resolution inference rule by lifting the condition of resolving only complementary literals. In this work, we use an LLM as the theory that identifies the unsatisfiable natural language predicates to do reasoning via theory resolution.

Natural Language Logic Natural language encompasses a significant amount of information that cannot be easily represented using symbolic logic. Although one can represent functions and predicates in symbolic logic, it may be hard to fully axiomatize their real-world meaning, which is a substantial limitation of the semantic parsing approaches. For instance, being *"spicy"* and having *"a kick to it"* are assigned completely different predicates, and pure symbolic reasoning cannot identify the intuitive entailment relationship between them without specific axioms. Moreover, representing commonsense knowledge in symbolic logic is very challenging (Davis, 2014). However, LLMs are capable of understanding the semantic relationship between such predicates and also encompass substantial commonsense knowledge, which can be used for reasoning in real-world applications.

As mentioned earlier, theory resolution offers the capability to resolve non-complementary literals if they are deemed unsatisfiable by a theorem prover. By employing an LLM as the theorem prover, we can leverage the theory resolution framework to conduct resolution within an extended version of First-Order Logic, where predicates and functions are no longer symbols but natural language texts, a system we call *Natural Language (NL) Logic*.

Using the LLM theorem prover in the NL logic, the unsatisfiability condition of the theory resolution reduces to natural language entailment. In other words, if an LLM identifies a natural language predicate B to entail predicate D, i.e., $B(x) \vdash_{LLM} D(x)$, and therefore, $B(x) \wedge \neg D(x) \vdash_{LLM} \perp$, then literals $B(x)$ and $D(x)$ can be resolved. For instance, given clauses $c_1 =$ *"kick to it"*(x) and $c_2 = \neg$*"spicy"* $(x) \vee Q(x)$, in which $Q(x)$ is another literal with a natural language predicate, since the LLM identifies the natural language entailment *"kick to it"* $\vdash_{LLM}$ *"spicy"*, a theory resolution step can be performed as

$$\frac{\text{"kick to it"}(x) \quad \neg\text{"spicy"}(x) \vee Q(x)}{Q(x)}. \tag{3}$$

3.1 LLM-TRes Algorithm

This section presents LLM-TRes, an algorithm for efficient logical commonsense reasoning based on theory resolution using LLMs. The workflow of LLM-TRes is shown through a worked example in Figure 1, and formalized in Algorithm 1.

Problem Definition Consider a set of queries $\mathcal{Q}$ and a knowledge base (KB) denoted by $\mathcal{K}$ which comprises a set of axioms $\mathcal{A}$ and a set of facts $\mathcal{F}$, all represented in natural language logic in clausal form. In this work, we aim to propose an inference rule i that for each $q \in Q$, finds a set of proofs denoted by *proofs*, such that each proof $f \in$ *proofs* consists of a subset of clauses in $\mathcal{K}$, and is assigned a priority score ρ reflecting the priority of the proof.

Algorithm To prove that $\mathcal{K}$ entails the query q via resolution, we must demonstrate that iteratively applying resolution to derive new clauses from $\mathcal{K} \wedge \neg q$ leads to a contradiction, thereby proving its unsatisfiability. The first question that arises is which clause should be chosen to begin the resolution proof. Two major paradigms are used in performing resolution: (i) starting from the clauses in $\mathcal{K}$ to derive q from them and resolve it with $\neg q$, an approach known as *forward chaining*, or (ii) starting from $\neg q$ and resolving it with clauses in $\mathcal{K}$ to reach a contradiction, known as *backward chaining*. Since backward chaining employs a goal-driven approach, which is shown to improve efficiency in reasoning over natural language (Kazemi et al., 2023), we begin the resolution process from $\neg q$. Therefore, $\neg q$ becomes our first *active clause* that we need to resolve with a clause from $\mathcal{K}$.

The potentially enormous size of $\mathcal{K}$ poses a major challenge. Also, as the resolution process progresses, new clauses are created, leading to a further expansion in the size of the search space. To perform resolution efficiently in this combinatorial search space, LLM-TRes employs two strategies: (i) prioritizing the resolvent clauses to continue the resolution process, and (ii) restricting the theory resolution search space using semantic similarity.

Resolution Priority Definition and Ordering: The first mechanism employed in LLM-TRes to enable efficient resolution is prioritizing candidate clauses. Using this prioritization scheme, resolvent clauses that have a higher potential for being part of a plausible proof will be given precedence over clauses generated from less plausible resolution steps. The plausibility of a theory resolution step, in which an active clause c is resolved with a clause c_{target} to generate the resolvent clause c_{res}, denoted by $\rho^{entail}_{c_{res}}$, is determined by calculating the probability that the LLM assigns to c_{target} entailing c.

$$\rho^{entail}_{c_{res}} = p(c_{target} \vdash_{LLM} c). \tag{4}$$

These plausibility scores can help us prioritize the resolvent clauses. For instance, in the example provided in Figure 1, since resolving *"shrimp"* with *"seafood"* yields a higher entailment score than resolving *"garlic"* with *"seafood"*, it is intuitive to prioritize the first resolvent as it is more likely to be part of the final proof. Since we are interested in identifying the most plausible proofs, i.e., the sequences of theory resolution steps with the highest entailment scores, we define the first entry of our priority score for each resolvent clause c_{res} as the overall entailment score of all resolution steps beginning from the original negated query that led to its derivation. Denoting the set of parent clauses of c_{res} as $\mathcal{P}_{c_{res}}$, we can inductively define the overall proof entailment score of c_{res} as

$$\rho^e(c_{res}) = \left(\prod_{c' \in \mathcal{P}_{c_{res}}} \rho^e(c') \right) \cdot \rho^{\text{entail}}_{c_{res}}. \tag{5}$$

Algorithm 1 LLM-TRes Algorithm

Input: $\mathcal{K}$, q, *max_proofs*, *max_iters*, b
proofs $\leftarrow \emptyset$
$PQ \leftarrow \emptyset$ // PQ is an initially empty priority queue.
$PQ.push(\neg q, (1, 0))$ // Negation of the initial query q has priority $(1, 0)$, PQ is ordered by Equation 7
while $i <$ *max_iters* **do**
  while $PQ \neq \emptyset \wedge i <$ *max_proofs* **do**
    $c \leftarrow PQ.pop()$
    if $c = \bot$ **then**
      max_proofs++
      proofs $\leftarrow$ *proofs* $\cup \{c\}$
    else
      $\beta_c \leftarrow b$ most likely candidates in $\mathcal{K}$ to resolve with c
      for $c_{\text{target}} \in \beta_c$ **do**
        Compute resolvent c_{res} of c and c_{target} using Equation 2
        $PQ.push(c_{\text{res}}, (\rho^e(c_{\text{res}}), \rho^l(c_{\text{res}}))$ // *cf. Equations 5 and 6*
Output: *proofs*

When choosing between equally plausible proofs, we are interested in shorter proofs that avoid redundant steps. We assign a second priority score to reflect this preference which is considered only when the entailment proof scores are equal. As for the proof entailment score, we can obtain the proof length score of c_{res} inductively from the maximum proof length of its parent clauses as

$$\rho^l(c_{res}) = 1 + \max_{c' \in \mathcal{P}_{c_{res}}} \rho^l(c'). \tag{6}$$

The final priority score for each resolvent clause c_{res} is formed as the tuple $(\rho^e(c_{res}), \rho^l(c_{res}))$ and all resolvents are pushed to a priority queue PQ. A total order of clauses in PQ is then determined as

$$\begin{aligned} c_1 \preceq c_2 \iff & [\rho^e(c_1) > \rho^e(c_2)] \\ & \vee [(\rho^e(c_1) = \rho^e(c_2)) \wedge (\rho^l(c_1) < \rho^l(c_2))]. \end{aligned} \tag{7}$$

Restricting Theory Resolution with Embeddings: The knowledge base may contain various axioms and facts, many of which are irrelevant to the active clause. To enhance efficiency and maintain the growth of the resolution space tractable, we restrict the size of our resolution search space by a branching factor b and select candidate target clauses for performing resolution based on their semantic relevance to the current active clause. Concretely, we use the similarity scores between $\mathbf{z}_c$, the word embedding vector of the active clause c, and $\mathbf{z}_{c'}$, word embedding vectors of each candidate clause c' to find β_c, the set of b most relevant clauses to c as

$$\beta_c = \{c' | (c' \in \mathcal{K}) \wedge (c' \neq c) \wedge (\mathbf{z}_c^T \mathbf{z}_{c'} \geq \tau)\}, \quad (8)$$

in which τ is set to the bth highest inner product score between embeddings of c and other clauses, thus resulting in top-b theory resolution candidates. Next, theory resolution can be performed between c and each clause in β_c as in Equation 2.

These two mechanisms together enable an efficient inference via LLM-based theory resolution. At the beginning of each iteration of LLM-TRes, the clause holding the foremost position in PQ becomes the active clause. Once a resolution step leads to contradiction, the proof and its respective priority score are added to the set of found proofs by backtracing the ancestor clauses up to the query.

This algorithm continues until either a certain number of proofs are found or the maximum number of iterations is exceeded. Notably, LLM-TRes is not limited to proving a single query; instead, it finds a set of proofs with each assigned a strength score. This functionality allows it to assess the likelihood of each query being entailed, which is beneficial for applications requiring ranking, such as answering multiple-choice questions. In applications where a binary truth value is considered for the query, the proof scores of q and $\neg q$ are compared. Our experiments cover both cases.

4 Repairability of Erroneous Resolution

Since LLM-TRes provides access to atomic inference steps in the resolution process, it facilitates verifiability and debuggability. Although the entailment probabilities assigned by the LLMs may be erroneous, the exact resolution step at which the failure occurs is discernible. Furthermore, it can be easily corrected by introducing a rectifying rule into the knowledge base. 1 presents an example. The LLM's mistake in assigning a low entailment score for *"catfish"* to entail *"seafood"* leads to incorrect reasoning. However, introducing the correct axiom $\forall y \text{"catfish"}(y) \implies \text{"seafood"}(y)$ to the KB repairs this mistake. The following proposition formalizes this property and is proven in Appendix A.

Proposition 1. *Consider proof P_c^ϕ using axiom ϕ that derives clause c. For any incorrect LLM reasoning axiom ϕ, a Repair Axiom ϕ' can be inserted such that $P_c^{\phi'}$ will be produced before P_c^ϕ.*

5 Experiments

We evaluate LLM-TRes on three different tasks involving commonsense reasoning on natural language data: preference reasoning, deductive reasoning, and causal commonsense reasoning. We release our implementation and data[1].

5.1 Tasks and Datasets

Preference Reasoning Understanding user preferences from natural language statements is a complex but essential task in applications requiring personalization. For this task, we use Recipe-MPR (Zhang et al., 2023a), a dataset consisting of 500 user queries stating their preference toward recipes, e.g., *"I would like meat lasagna but I'm watching my weight"* with five-way recipe options that requires a range of commonsense inference including temporal and analogical reasoning.

Deductive Reasoning We use ProntoQA (Saparov and He, 2022), a widely used dataset for evaluating the deductive reasoning ability of LLMs. This dataset consists of natural language queries about KBs including facts and axioms generated from ontologies. We use 500 queries of the true ontology as they are consistent with the real world and are useful to evaluate commonsense reasoning. We select the most challenging 5-hop subset of the dataset.

Causal Commonsense Reasoning We use COPA-SSE (Brassard et al., 2022), a dataset for reasoning about event causes and effects using a semi-structured KB. In the *"effect"* split of this dataset, an event is provided such as *"The pen ran out of ink."*, together with semi-structured explanations with assigned quality scores, and the task is to determine the more plausible candidate effect, e.g., *"I used a pencil."* or *"I signed my name."*.

5.2 Baselines and Evaluation

We use LAMBADA[2], a seminal work in formal reasoning with LLMs, and zero-shot Chain-of-Thought (CoT) prompting (Kojima et al., 2022) as our comparison baselines. Semantic parsing methods are inherently unable to perform commonsense reasoning and do not apply to our tasks. We use

[1] https://anonymous.4open.science/r/LLM-TRes-678C/

[2] Since the original paper did not release code, we use the implementation in (Lee and Hwang, 2024).

Table 1: Reasoning performance of methods across the three datasets. Gemma fails to provide explanations for Recipe-MPR, so reasoning scores cannot be calculated for it (Fail). LAMBADA requires a rule set that is not provided in Recipe-MPR, and cannot rank proofs which is necessary for COPA-SSE. Pure entailment does not generate proofs, so the reasoning scores do not apply to it (NA).

Method	Recipe-MPR			ProntoQA			COPA-SSE		
	Accuracy	RS Macro	RS Micro	Accuracy	RS Macro	RS Micro	Accuracy	RS Macro	RS Micro
CoT (GPT-3.5-Turbo)	**0.844**	0.850	0.900	0.738	0.600	0.878	0.860	0.800	0.921
CoT (Llama3 8B)	0.768	0.550	0.742	0.718	0.250	0.802	0.839	0.800	0.916
CoT (Gemma 7B)	0.460	Fail	Fail	0.588	0.250	0.657	0.818	0.520	0.450
CoT (Mistral 7B)	0.689	0.850	0.946	0.902	0.600	0.890	0.643	0.850	0.892
LAMBADA (GPT-3.5-Turbo)	NA	NA	NA	0.580	0.800	0.900	NA	NA	NA
Pure Entailment (BART 406M)	0.682	NA	NA	0.740	NA	NA	0.825	NA	NA
LLM-TRes (BART 406M)	0.822	**1.000**	**1.000**	**0.990**	**1.000**	**1.000**	**0.888**	**0.900**	**0.958**

GPT-3.5 Turbo as the LLM for LAMBADA and for converting the natural language axioms and queries to the clausal form in our method, and obtain the entailment probabilities for theory resolution using BART large (Lewis et al., 2020) model[3] trained on MNLI (Williams et al., 2018) dataset. We compare against CoT prompting with GPT-3.5 Turbo, Llama3 8B, Mistral 7B, and Gemma 7B. To ensure that the difference in the performance of our model and the baselines is not due to using different LLMs, we also use pure BART-large entailment scores between facts and query as a baseline.

We evaluate the reasoning performance of the models considering the correctness of the final answers, measured by the accuracy, and correctness of the reasoning process measured by the reasoning score (RS) which we manually assess for the first 20 queries the models answer correctly. RS is commonly evaluated as a binary judgment on whether the predicted proof is supported by the ground truth proof (Kazemi et al., 2023; Lee and Hwang, 2024). However, RS does not assess the number of errors. Therefore, in addition to this metric which we call *macro RS*, to provide a more fine-grained evaluation of the provided proofs, based on the idea provided in Min et al. (2023), we use a new metric which we name *micro RS*. Given a provided proof P and the ground truth proof P^*, and denoting the indicator function as $\mathbb{I}$, the micro RS for each query is defined as $RS_{Micro} = \frac{1}{|P|}\sum_{p\in P}\mathbb{I}(p \in P^*)$.

5.3 Results

RQ1: Comparison of Reasoning Performance Results of the reasoning performance are provided in Table 1. On deductive and causal commonsense reasoning tasks, LLM-TRes achieves higher accuracies than the baselines although the language models they use are multiple times larger. On preference reasoning, LLM-TRes achieves the second-highest accuracy after CoT prompting with GPT3.5 Turbo with a rather small margin. On ProntoQA, since the high-quality conversion of the query and the knowledge base to the clausal format is straightforward, LLM-TRes can prioritize complementary literals to perform exact resolution, resulting in a near-ideal performance. The failure cases of LLM-TRes are due to the LLM's limitation in understanding contraposition as noted in previous work (Zhang et al., 2024). Nonetheless, LLM-TRes maintains consistently high performance, unlike other baselines which vary across tasks. For instance, while CoT with GPT-3.5 excels on Recipe-MPR and COPA-SSE, it is outperformed by Mistral on ProntoQA, which in turn performs rather poorly on Recipe-MPR and COPA-SSE. On RS, LLM-TRes outperforms all baselines across the three datasets at both the macro and micro level, showcasing its capability to provide proofs that are consistent with the ground truth proof. LAMBADA is unable to reason on Recipe-MPR as it performs backward chaining on explicit rule sets, which Recipe-MPR does not provide. Also, since LAMBADA can only prove or refute a query based on a KB and cannot score and rank the plausibility of proofs, it cannot choose the more plausible effects on the COPA-SSE dataset. Since CoT using Gemma refrained from providing any proof for preference reasoning despite being prompted to do so, the reasoning score could not be calculated for it. Finally, pure entailment does not provide proofs so RS cannot be evaluated.

RQ2: Robustness to Incompleteness of the KB Assuming access to a complete KB in which all required axioms are provided is often unrealistic

[3] https://huggingface.co/facebook/bart-large-mnli

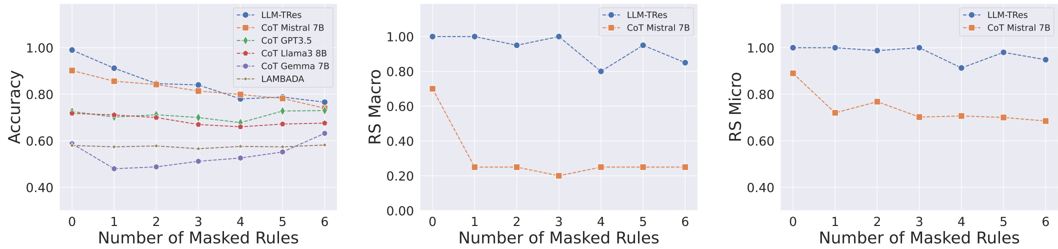

Figure 2: Reasoning performance of different models on ProntoQA with an incomplete KB. We mask out a number of rules to vary the degree of incompleteness of KB.

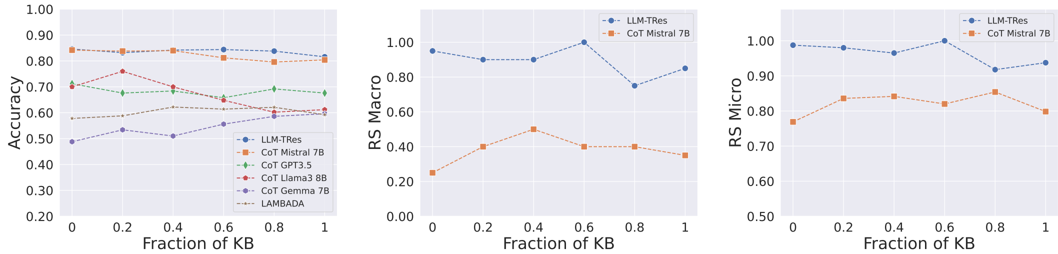

Figure 3: Reasoning performance of different models on ProntoQA with larger KB. We sample random axioms from other queries to increase the size of KB.

in practical applications. Therefore, a commonsense reasoning methodology must be able to extract the intrinsic commonsense knowledge of the LLMs to overcome the incompleteness of the KB. We assess this capability by repeatedly running experiments on ProntoQA each time removing a number of randomly chosen rules from the KB. We chose ProntoQA for this study as it is the only dataset with large rule sets that enables experiments with various ablated rules. Results of this experiment are provided in Figure 2. Although ablating rules from the KB decreases the accuracies of both LLM-TRes and the best baseline, CoT with Mistral, LLM-TRes often maintains higher accuracy. Moreover, the consistently higher reasoning score of LLM-TRes proves its superior ability to generate valid proofs.

RQ3: Robustness to Increase in Size of the KB In this experiment, we evaluate the robustness of LLM-TRes and other baselines to increases in the KB size. We form a large KB consisting of 75 distinct rules across different ProntoQA queries and each time add a fraction of this KB to the original rule set of the query while randomly masking 2 rules of the original KB. This experiment mainly aims to determine if the restricting resolution search space of LLM-TRes using semantic similarity can identify the relevant clauses to the active clause. In all tests, LLM-TRes uses the similarity between GPT-3 embeddings of the clauses with a branching factor of 15. Meanwhile, other baselines include the entire KB in the prompt which is costly and inefficient. Results of this test, shown in Figure 3 depict that LLM-TRes and the best baseline, CoT with Mistral, sustain their performance, but LLM-TRes consistently obtains higher reasoning scores while using a more efficient methodology for pruning the reasoning search space.

6 Conclusion

We presented LLM-TRes, a novel framework for formal reasoning with LLMs based on theory resolution that allowed us to integrate LLMs into resolution logical reasoning seamlessly. By providing access to every atomic reasoning step, LLM-TRes enabled *verifiability* and *debuggability* of the process. It also offered a reliable *repairing* mechanism for correcting flaws in the LLM reasoning by asserting the particular missed axiom which was theoretically guaranteed to override the mistakenly low-probability resolution step. The promising performance of LLM-TRes on preference reasoning, deductive reasoning, and causal commonsense reasoning tasks demonstrates its efficacy in providing accurate answers and correct proofs. These capabilities make LLM-TRes a robust foundation for counteracting hallucination and pave the way for more trustworthy deployment of LLM-based commonsense reasoners in applications where correctness, verifiability, and repairability are paramount.

Limitations

While we believe this work has made substantial progress in verifiable, debuggable, and repairable commonsense reasoning, it naturally has limitations that we hope will encourage further investigation and future work. As mentioned in the paper, we provided a reliable mechanism for erroneous reasoning processes; however, determining a flawed step requires expert judgment. In our work, we do not focus on evaluating the reasoning steps and how the repair axioms are introduced. Proposing an automated mechanism for evaluating the reasoning steps can be a direction of future research. Furthermore, as in all LLM-based reasoning methodologies, obtaining high reasoning performances requires an apt LLM. As we discussed in Section 5.3, limitations of the utilized LLM such as their shortcomings in understanding contraposition can pose challenges to the overall performance of the method. Finally, as we mentioned in the paper, LLM-TRes focuses on the natural language extension of First Order Logic (FOL) which we introduced, and extending it to Higher-Order Logic (HOL) could be considered as a future research direction given the prior uses of HOL in formalizing natural language semantics and complex modal constructs (van Eijck and Unger, 2010).

Ethics Statement

Our contribution of LLM-TRes aims to enable transparent reasoning with LLMs such that correctness of every reasoning step can be verified and potentially repaired if incorrect. However, it is important for us to note that a correct proof or line of argument from premises neither presupposes that the premises are ethical nor that the conclusion derived from the premises and line of reasoning is ethical. In this sense, practical use of LLM-TRes still requires ethical oversight to monitor ethical and bias considerations for any axioms entered by the user as well as to verify that unintended reasoning hallucinations by the underlying LLM have not led to incorrect, biased, or unethical conclusions.

References

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556.

Peter Baumgartner. 1992. An ordered theory resolution calculus. In *Logic Programming and Automated Reasoning: International Conference LPAR'92 St. Petersburg, Russia, July 15–20, 1992 Proceedings 3*, pages 119–130. Springer.

Ning Bian, Xianpei Han, Le Sun, Hongyu Lin, Yaojie Lu, Ben He, Shanshan Jiang, and Bin Dong. 2023. Chatgpt is a knowledgeable but inexperienced solver: An investigation of commonsense problem in large language models. *arXiv preprint arXiv:2303.16421*.

Ana Brassard, Benjamin Heinzerling, Pride Kavumba, and Kentaro Inui. 2022. Copa-sse: Semi-structured explanations for commonsense reasoning. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3994–4000.

Chin-Liang Chang and Richard Char-Tung Lee. 2014. *Symbolic logic and mechanical theorem proving*. Academic press.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*.

Ernest Davis. 2014. *Representations of commonsense knowledge*. Morgan Kaufmann.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

Kathleen M Galotti. 1989. Approaches to studying formal and everyday reasoning. *Psychological bulletin*, 105(3):331.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2023. Lambada: Backward chaining for automated reasoning in natural language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6547–6568.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Jinu Lee and Wonseok Hwang. 2024. Symba: Symbolic backward chaining for multi-step natural language reasoning. *arXiv preprint arXiv:2402.12806*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100.

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

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

Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. 2023. Large language models are competitive near cold-start recommenders for language-and item-based preferences. In *Proceedings of the 17th ACM conference on recommender systems*, pages 890–896.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. 2024. Testing the general deductive reasoning capacity of large language models using ood examples. *Advances in Neural Information Processing Systems*, 36.

Murray Shanahan. 2024. Talking about large language models. *Communications of the ACM*, 67(2):68–79.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. 2023. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*.

Mark E Stickel. 1985. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1(4):333–355.

Jan van Eijck and Christina Unger. 2010. *Computational Semantics with Functional Programming*. Cambridge University Press.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Yudong Xu, Wenhao Li, Pashootan Vaezipoor, Scott Sanner, and Elias B Khalil. 2023. Llms and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations. *arXiv preprint arXiv:2305.18354*.

Haochen Zhang, Anton Korikov, Parsa Farinneya, Mohammad Mahdi Abdollah Pour, Manasa Bharadwaj, Ali Pesaranghader, Xi Yu Huang, Yi Xin Lok, Zhaoqi Wang, Nathan Jones, et al. 2023a. Recipe-mpr: A test collection for evaluating multi-aspect preference-based natural language retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2744–2753.

Yanfang Zhang, Yiliu Sun, Yibing Zhan, Dapeng Tao, Dacheng Tao, and Chen Gong. 2024. Large language models as an indirect reasoner: Contrapositive and contradiction for automated reasoning. *arXiv preprint arXiv:2402.03667*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023b. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2024. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Proof of Repairability of LLM-TRes

Proposition 2. *Consider proof P_c^ϕ using axiom ϕ that derives clause c. For any incorrect LLM reasoning axiom ϕ, a Repair Axiom ϕ' can be inserted such that $P_c^{\phi'}$ will be produced before P_c^ϕ.*

Proof. A proof $P_c^\phi = P_c \cup \{\phi\}$ can be viewed as the combined set of clauses P_c and ϕ that derive clause c. We can define the proof score $\rho^e(P_c^\phi)$ of clause c by inductively unrolling Equation 5 for $\rho^e(c)$ over all ancestor clauses P_c^ϕ that derive it. This yields a simple product form: $\rho^e(P_c^\phi) = \rho_\phi^{entail} \cdot \prod_{c' \in P_c} \rho_{c'}^{entail}$. Now, comparing two different derivations P_c^ϕ and $P_c^{\phi'}$ of c, we can easily show that $\rho^e(P_c^{\phi'}) > \rho^e(P_c^\phi)$ since $\frac{\rho^e(P_c^{\phi'})}{\rho^e(P_c^\phi))} = \frac{\rho_{\phi'}^{entail} \cdot \prod_{c' \in P_c} \rho_{c'}^{entail}}{\rho_\phi^{entail} \cdot \prod_{c' \in P_c} \rho_{c'}^{entail}} = \frac{\rho_{\phi'}^{entail}}{\rho_\phi^{entail}} > 1$ given that the explicit Repair Axiom has $\rho_{\phi'}^{entail} = 1$ (by definition) while the LLM entailment score $\rho_\phi^{entail} < 1$ (necessarily). Hence, the proof $P_c^{\phi'}$ containing the Repair Axiom ϕ' will always be given precedence over P_c^ϕ according to the total ordering of Equation 7 used to prioritize proofs in the LLM-TRes Algorithm 1. □

B Anecdotal Examples

To offer deeper insight into the responses and proofs generated by LLM-TRes and the comparison baselines, this section presents anecdotal examples illustrating each model's performance on the evaluated tasks. Specifically, we showcase the outputs from the following models:

- LLM-TRes
- Chain of Thought prompting with Mistral
- Chain of Thought prompting with Llama3
- Chain of Thought prompting with Gemma
- Chain of Thought prompting with GPT-3.5
- LAMBADA

We apply these models to three distinct tasks, offering a comparative analysis of their responses. Detailed anecdotal examples are provided to elucidate the models' capabilities and limitations. In each anecdotal example, we present the query, the rule set (on datasets that contain it), and the ground truth proof, followed by the response that each model provides. We also indicate correct proof steps with green color and highlight incorrect ones in red.

An example of the Deductive reasoning task, as evaluated on the ProntoQA dataset, is detailed in Section B.1. Additionally, we explore the models' performance on causal commonsense reasoning using the COPA-SSE dataset and on preference reasoning using the Recipe-MPR dataset, with examples discussed in Sections B.2 and B.3, respectively.

B.1 Deductive Reasoning

Query: is x imaginary?
Rule set: Real numbers are numbers. Every real number is not imaginary. Each integer is a real number. Natural numbers are integers. Each natural number is not negative. Every prime number is a natural number. Each prime number is prime. Mersenne primes are prime numbers. Every Mersenne prime is not composite. Complex numbers are imaginary. x is a Mersenne prime.
Ground Truth Proof: x is a Mersenne prime. Mersenne primes are prime numbers. Every prime number is a natural number. Natural numbers are integers. Each integer is a real number. Every real number is not imaginary. Answer: False.

LLM-TRes:

Proof for query *imaginary(x)*:
Negated query: $\neg$*imaginary(x)*
Step 1-

$$\frac{\neg \textit{imaginary(x)} \quad \neg \textit{complex(x)} \vee \textit{imaginary(x)}}{\neg \textit{complex(x)}}\ .$$

Step 2-

$$\frac{\neg \textit{complex(x)} \quad \neg \textit{real(x)} \vee \neg \textit{imaginary(x)}}{\neg \textit{real(x)}}\ .$$

Step 3-

$$\frac{\neg \textit{real(x)} \quad \neg \textit{integer(x)} \vee \textit{real(x)}}{\neg \textit{integer(x)}}\ .$$

Step 4-

$$\frac{\neg \textit{integer (x)} \quad \neg \textit{natural number(x)} \vee \textit{integer(x)}}{\neg \textit{natural number(x)}}\ .$$

Step 5-

$$\frac{\neg \textit{natural number(x)} \quad \neg \textit{prime number(x)} \vee \textit{natural number(x)}}{\neg \textit{prime number(x)}}\ .$$

Step 6-

$$\frac{\neg \textit{prime number(x)} \quad \neg \textit{mersenne prime number(x)} \vee \textit{prime number(x)}}{\neg \textit{mersenne prime number(x)}}\ .$$

Step 7-

$$\frac{\neg \textit{mersenne prime number(x)} \quad \textit{mersenne prime number(x)}}{\bot}\ .$$

Proof score: $\rho_{\text{query}} = (0.923, 7)$

* * *

Proof for negation of query $\neg$*imaginary(x)*:
Negated query: *imaginary(x)*
Step 1-

$$\frac{\textit{imaginary(x)} \quad \neg \textit{real(x)} \vee \neg \textit{imaginary(x)}}{\neg \textit{real(x)}}\ .$$

Step 2-

$$\frac{\neg \textit{real(x)} \quad \neg \textit{integer(x)} \vee \textit{real(x)}}{\neg \textit{integer(x)}}\ .$$

Step 3-

$$\frac{\neg integer\ (x) \quad \neg natural\ number(x) \lor integer(x)}{\neg natural\ number(x)}.$$

Step 4-

$$\frac{\neg natural\ number(x) \quad \neg prime\ number(x) \lor natural\ number(x)}{\neg prime\ number(x)}.$$

Step 5-

$$\frac{\neg prime\ number(x) \quad \neg mersenne\ prime\ number(x) \lor prime\ number(x)}{\neg mersenne\ prime\ number(x)}.$$

Step 6-

$$\frac{\neg mersenne\ prime\ number(x) \quad mersenne\ prime\ number(x)}{\bot}.$$

Proof score: $\rho_{\neg \text{query}} = (1, 6)$

* * *

Since $\rho^e_{\neg \text{query}} > \rho^e_{\text{query}}$, the answer is False.

CoT with Mistral: 1. Rule 1: Every real number is not imaginary.
2. Rule 11: Complex numbers are imaginary.
3. Given:x is a Mersenne prime.
4. Rule 1: Every Mersenne prime is a real number (since it is a prime number and a prime number is a real number).
5. Since x is a Mersenne prime and a real number, it is not imaginary by definition.
Therefore, the answer to the query is False.

CoT with Llama3: From the Knowledge Base, we know that "x is a Mersenne prime" (Rule 9).
From Rule 8, we know that "Every Mersenne prime is not composite".
From Rule 7, we know that "Each prime number is prime".
From Rule 6, we know that "Every prime number is a natural number".
From Rule 5, we know that "Each natural number is not negative".
From Rule 4, we know that "Natural numbers are integers".
From Rule 3, we know that "Each integer is a real number".
From Rule 2, we know that "Every real number is not imaginary".
From Rule 1, we know that "Real numbers are numbers".
Since x is a Mersenne prime, and Mersenne primes are not composite, and prime numbers are natural numbers, and natural numbers are integers, and integers are real numbers, and real numbers are not imaginary, we can conclude that x is not imaginary. Therefore, the answer to the query is False.

CoT with Gemma: I used the following rules from the Knowledge Base to answer the query:
"Real numbers are numbers"
"Every real number is not imaginary"
"Mersenne primes are prime numbers"
"Every Mersenne prime is not composite"
Answer: Therefore, the answer to the query is False.

CoT with GPT3.5: To answer this query, we will use the following rules from the Knowledge Base: 1. Each prime number is prime. 2. Each prime number is not composite. 3. x is a Mersenne prime. 4. Every Mersenne prime is not composite. 5. Complex numbers are imaginary.
Proof: From rule 3 and rule 4, we can conclude that x is not a composite number. Since x is a Mersenne prime, and every Mersenne prime is not composite (rule 4), we can infer that x is a prime number. From rule 1, "Each prime number is prime". Finally, from rule 5, "Complex numbers are imaginary", we can conclude that if x is a prime number and all prime numbers are prime, then x is not imaginary.
Therefore, the answer to the query is False.

LAMBADA: Begin proof from "x is imaginary."
Fact selection: No fact proves this rule.

Goal decomposition by selecting the next rule: "Complex numbers are imaginary."
Goal decomposed to: x is a complex number.
Fact selection: found fact: Every Mersenne prime is not composite.
Fact verification failed. (the fact does not prove the subgoal.)
Goal decomposition by selecting the next rule: Complex numbers are imaginary.
Goal decomposed to: x is an imaginary number.
Fact selection: found fact: x is a Mersenne prime.
Fact verification failed. (the fact does not prove the subgoal.)
Goal decomposition by selecting the next rule: Complex numbers are imaginary.
Goal decomposition failed.
Since the query cannot be proven, it is False.

B.2 Causal Commonsense Reasoning

Query: The crowd intensified.
Options: 1- The father handed his son some money., 2- The father grabbed his son's hand.
Answer: The father grabbed his son's hand.
Rules: 1- (an intense crowd, capable of, making your son hard to find), 2- (making your son hard to find, causes desire, keep an eye on son), 3- (keeping an eye on son, results in, grabbing son's hand), 4- (handing money to son, causes, son having money), 5- (father, capable of, handing money to son)
Ground Truth Proof: 1, 2, 3.

LLM-TRes:

Proof for Option 1:
Negation of query: $\neg$*handing his son money(father)*
Step 1-

$$\frac{\neg \textit{handing his son money(father)} \quad \neg \textit{making son hard to find(x)} \vee \textit{keep an eye on son(x)}}{\neg \textit{making son hard to find(man)}} \; \theta = \textit{[x/father]}.$$

Step 2-

$$\frac{\neg \textit{making son hard to find(x)} \quad \neg \textit{intense crowd(x)} \vee \textit{could make finding son hard(x)}}{\neg \textit{intense crowd(x)}} \; \theta = \textit{[x/father]}.$$

Step 3-

$$\frac{\neg \textit{instense crowd(x)} \quad \textit{intensified crowd(x)}}{\bot} .$$

Proof score: $\rho_{\text{option1}} = (0.005, 3)$

* * *

Proof for Option 2:
Negation of query: $\neg$*grabbed son's hand(father)*
Step 1-

$$\frac{\neg \textit{grabbed son's hand(father)} \quad \neg \textit{making son hard to find(x)} \vee \textit{keep an eye on son(x)}}{\neg \textit{making son hard to find(man)}} \; \theta = \textit{[x/father]}.$$

Step 2-

$$\frac{\neg \textit{making son hard to find(x)} \quad \neg \textit{intense crowd(x)} \vee \textit{could make finding son hard(x)}}{\neg \textit{intense crowd(x)}} \; \theta = \textit{[x/father]}.$$

Step 3-

$$\frac{\neg \textit{instense crowd(x)} \quad \textit{intensified crowd(x)}}{\bot} .$$

Proof score: $\rho_{\text{option2}} = (0.883, 3)$

* * *

Since $\rho^{e}_{\text{option2}} > \rho^{e}_{\text{option1}}$, the answer is Option2.

CoT with Mistral: Based on the given knowledge base, the consequence "The father grabbed his son's hand" is the more probable one. Proof: According to the knowledge base, an intense crowd can make your son hard to find. This situation would likely cause a desire in the father to keep a close eye on his son (rule: ["making your son hard to find", "causes desire", "keep an eye on son"]). The rule ["keeping an eye on son", "results in", "grabbing son's hand"] states that keeping an eye on his son results in the father grabbing his son's hand. Therefore, the father grabbing his son's hand is a more probable consequence given the knowledge base. Additionally, the rule ["father", "capable of", "handing money to son"] states that a father is capable of handing money to his son. However, there is no rule directly linking an intense crowd to the father handing money to his son. Therefore, the probability of the father handing his son some money is lower than the father grabbing his son's hand.
CoT with Llama3: Consequence: The father handed his son some money. Proof: The rules used from the Knowledge Base are: 1. ["father", "CapableOf", "handing money to son"] 2. ["handing money to son", "Causes","son having money"] From the event "The crowd intensified", we can infer that the crowd was intense, which is related to the first rule. This rule states that a father is capable of handing money to his son. The second rule states that handing money to son causes son having money. Therefore, we can conclude that the father handed his son some money.
CoT with Gemma: The father handed his son some money. (Gemma did not provide any proof for this query.)
CoT with GPT3.5: The father handed his son some money. Proof: 1. "handing money to son", "causes", "son having money" (KB) 2. "father", "capable of", "handing money to son" (KB) 3. Therefore, the father handing money to his son can be inferred from the KB.

B.3 Preference Reasoning

Query: I'd like a seafood recipe with a kick to it. **Options**: 1- Garlic shrimp., 2- Cajun catfish stew. **Answer**: 2- Cajun catfish stew.
LLM-TRes: Please refer to Figure 1.
CoT with Mistral: The query asks for a seafood recipe with a kick to it. Option 1, Garlic shrimp, is a classic seafood recipe that is known for its strong garlic flavor, but it doesn't necessarily have a "kick" to it. Option 2, Cajun catfish stew, on the other hand, is a spicy seafood dish that is known for its bold and fiery flavors, which would provide the desired kick. Therefore, the selected option is: 2. Cajun catfish stew.
CoT with Llama3: I think the most suitable option for a seafood recipe with a kick is the Cajun catfish stew because it combines the bold flavors of Cajun cuisine with the richness of seafood, which is likely to provide a spicy and savory experience. Therefore, the selected option is: 2- Cajun catfish stew.
CoT with Gemma: The query calls for a seafood recipe with a kick. Garlic shrimp is a popular seafood recipe that is known for its bold flavor and spicy kick. Therefore, the selected option is: 1- Garlic shrimp.
CoT with GPT3.5: The query asks for a seafood recipe with a kick, which implies a dish with bold and spicy flavors. While garlic shrimp is delicious, Cajun cuisine is known for its spicy and flavorful dishes, making Cajun catfish stew a more suitable option for someone looking for a seafood recipe with a kick., the selected option is: 2. Cajun catfish stew.

C Prompts Used for Language Models

In this section, we provide prompts used in performing experiments with language models in this paper. They are also included in the repository containing our code and data.

Prompt for Deductive Reasoning with Baselines

```
Task: You will be given a query about an
    object x. You are a helpful and smart
    assistant trying to answer this query.
    In order to do this, a fact about x
    and a set of rules are provided to you
    in a Knowledge Base. Using these
    rules, you must both provide an answer
    to the query (the answer has to be
    "True" or "False") and give a proof of
    your answer by using the rules from
    Knowledge Base. Think step-by-step and
    try to use the rules one-by-one to
    answer the query. Begin your response
    by providing the proof and stating the
    rules you used from the knowledge base
    to give the answer. Then, give your
    final answer to the query by saying
    either "Therefore, the answer to the
    query is True" or "Therefore, the
    answer to the query is False" and not
    saying anything else.
Query: {{ QUERY }}
KB: {{ KB }}
```

Prompt for Causal Commonsense Reasoning with Baseline LLMs

```
Task: You will be given a sentence
    about an event. Also, a number of
    rules in the form of a Knowledge
    Base are presented to you. For
    this event, two possible
    consequences are given. You need
    to determine which of these
    consequences can be inferred from
    the event and the rules in the
    Knowledge Base. You must provide a
    proof for your answer by using the
    rules from the Knowledge Base.
    First, copy the consequence you
    think can be inferred. Then, in
    the next line, provide your proof
    by stating the rules you used from
    the Knowledge Base. Let's think
    step by step.

Event: {{ EVENT }}
KB: {{ KB }}
Consequence1: {{ CONSEQUENCE1 }}
Consequence2: {{ CONSEQUENCE2 }}
```

Prompt for Preference Reasoning with Baseline LLMs

```
Task: Consider the provided query and the
    set of options. You must pick the
    option that is most suitable for the
    query. Think step by step. First,
    explain your reason for why you
    think this recipe is the most
    proper. Remember that you have to
    state the reason first. Then,
    mention the most proper recipe by
    saying: Therefore, the selected
    option is: <option number>.
Query: {{ QUERY }}
Options: {{ OPTIONS }}
```

Prompt for Conversion of Natural Language KB to Clausal Form

```
Task: you are a First-Order Logic expert.
    A sentence written in Natural
    Language will be presented to you.
    Convert that sentence to First-Order
    Logic. In this conversion, follow
    these syntactic rules:
1- Instead of universal quantifier, write
    FOR_ALL .
2- Write all predicates for the variable
    (x) , even if the sentence refers to
    a specific object. For example, "127
    is an integer" must be converted to
    "integer(x)" or "Bob is a cat" must
    be converted to "cat(x)".
3- If the predicate name has multiple
    parts, use _ instead of       in the
    name.
4- Instead of the implication symbol, use
    => .
5- Use  ~  as the symbol of negation.
6- Only use lowercase letters for
    predicate names.
7- Even if the sentence is incorrect in
    your opinion, convert it to FOL
    given the stated rules without any
    further explanation.
8- If the sentence is not in the format
    of a universal statement, just state
    it as a predicate. For example, "Bob
    is a cat" must be converted to
    "cat(x)".
[few-shot examples]
```