Socrates or Smartypants: Testing Logic Reasoning Capabilities of Large Language Models with Logic Programming-Based Test Oracles

Zihao Xu[1*], Junchen Ding[1*], Yiling Lou[2], Kun Zhang[3], Dong Gong[1], Yuekang Li[1†]

[1]University of New South Wales, Australia
[2]Fudan University, China
[3]Carnegie Mellon University, USA
[1]{zihao.xu2, junchen.ding, dong.gong, yuekang.li}@unsw.edu.au, [2]yilinglou@fudan.edu.cn, [3]kunz1@cmu.edu

Abstract

Large Language Models (LLMs) have achieved significant progress in language understanding and reasoning. Evaluating and analyzing their logical reasoning abilities has therefore become essential. However, existing datasets and benchmarks are often limited to overly simplistic, unnatural, or contextually constrained examples. In response to the growing demand, we introduce SMARTYPAT-BENCH, a challenging, naturally expressed, and systematically labeled benchmark derived from real-world high-quality Reddit posts containing subtle logical fallacies. Unlike existing datasets and benchmarks, it provides more detailed annotations of logical fallacies and features more diverse data. To further scale up the study and address the limitations of manual data collection and labeling, such as fallacy-type imbalance and labor-intensive annotation, we introduce SMARTYPAT, an automated framework powered by logic programming-based oracles. SMARTYPAT utilizes Prolog rules to systematically generate logically fallacious statements, which are then refined into fluent natural-language sentences by LLMs, ensuring precise fallacy representation. Extensive evaluation demonstrates that SMARTYPAT produces fallacies comparable in subtlety and quality to human-generated content and significantly outperforms baseline methods. Finally, experiments reveal insights into LLM capabilities, highlighting that while excessive reasoning steps hinder fallacy detection accuracy, structured reasoning enhances fallacy categorization performance.

1 Introduction

LLMs demonstrate strong performance across diverse domains. As adoption increases, thorough evaluation across reasoning, domain knowledge, and problem-solving becomes crucial. Among these, logical reasoning is foundational, especially for tasks requiring structured thinking, such as programming. Prior studies explored LLMs' logical reasoning via symbolic-to-natural language conversion (Han et al. 2022; Parmar et al. 2024), but these often yield rigid, unnatural text using formal constructs ($\forall, \exists$) rarely seen in human language. To address this, Jin et al. (2022a) proposed the LOGIC dataset, short, realistic, but still simplistic fallacious statements from student quizzes labeled by fallacy type. Yet, many examples remain trivial. To address these limitations, researchers introduced the COIG-CQIA benchmark (M-A-P 2024), which features subtle logical errors drawn from forum posts on the Chinese platform *ruozhiba*. To adapt this content to English, Zhai et al. (2025) proposed a translated benchmark. However, its reliance on direct Chinese-to-English translation weakens context-sensitive nuances critical for evaluating logical reasoning. Moreover, the lack of annotated fallacy types limits the applicability for robust fallacy categorization.

A key challenge remains: building a benchmark that is challenging, derived from real-world English data, and annotated with specific fallacy types. We introduce SMARTYPAT-BENCH, derived from a Reddit community (Reddit 2025) analogous to the Chinese forum ruozhiba. We manually reviewed 2,500 posts (by upvotes), removing low-quality content and selecting 502 posts for annotation. Our dataset highlights two key desiderata: ❶ Controllable generation of subtle logical fallacies. Manual datasets are imbalanced, three fallacy types dominate 79.7%, while three rare ones comprise just 1.77%. Generation offers better balance and quality. ❷ Fidelity to intended fallacy types. Accurate generation eases labeling and improves benchmark reliability. To address these issues, we propose SMARTYPAT, an automated generator of logically fallacious statements for LLM evaluation. SMARTYPAT derives Prolog rules and fact structures from SMARTYPAT-BENCH, capturing the core logic of each fallacy type. LLMs then generate diverse fact instances conforming to these structures, which are verified and composed into natural language via Prolog, combining symbolic rigor with neural diversity and fluency, yielding a synthetic dataset, SMARTYPAT-BENCH-AUGMENTED.

SMARTYPAT produces high-quality, subtly fallacious statements comparable to human-written posts, outperforming two baselines: direct LLM generation and LLM-generated Prolog. We assess LLMs on fallacy detection and categorization. In the detection task, reasoning models tend to underperform due to overanalysis, leading to high FPR and lower F1 scores, while they generally excel at categorization.

In summary, we make the following contributions:

1. We present SMARTYPAT-BENCH, the first benchmark comprising 502 high-quality, real-world English statements labeled with fine-grained logical fallacy types.
2. We introduce SMARTYPAT, the first Prolog-based neural

*These authors contributed equally.
†Corresponding author

symbolic generation framework that synthesizes logically fallacious statements with built-in test oracles for evaluating LLM reasoning.
3. We conduct the first comprehensive evaluation of nine state-of-the-art LLMs on logical fallacy detection and classification, uncovering critical reasoning and alignment limitations.

We provide an extended version with source code and data links in (Xu et al. 2025). All appendices referenced in the paper correspond to this extended version.

2 Related Work

Testing Deep Learning Models. DeepGauge (Ma et al. 2018) pioneered testing deep learning (DL) systems, emphasizing the importance of test oracles for DL, including LLMs. Subsequent works adapted traditional software testing techniques, such as mutation testing (Humbatova, Jahangirova, and Tonella 2021) and fuzzing (Xie et al. 2019), to DL systems. Recently, logic programming was introduced for generating logically sound factual knowledge to test LLMs for hallucination detection (Li et al. 2024a), highlighting its potential for effective LLM testing.

Logical Fallacy Categorization Currently, there is no universally agreed-upon classification scheme for logical fallacies, and existing categorizations often include overlapping concepts. For instance, Li et al. (2025) introduce the category *Lame Jokes*, representing failures to grasp general knowledge or common sense, alongside *Factual Error*, which similarly involves misunderstandings of basic facts. These two categories substantially overlap, potentially leading to inconsistencies in annotation. Similarly, Zhai et al. (2025) define *Logical Error* as contradictions or flawed reasoning, *Commonsense Misunderstanding* as mistakes about everyday facts, and *Erroneous Assumption* as incorrect premises. However, all these definitions could reasonably fall under a broader category such as *Logical Error*, introducing ambiguity for both LLM interpretation and human annotation.

By analyzing existing classifications alongside the fallacious statements in our SMARTYPAT-BENCH dataset, we propose a refined categorization of logical fallacies with 14 distinct types, as shown in Table 2. Further details on this categorization are provided in the appendix.

3 A Preliminary Study

Limitations of Existing Benchmarks Researchers have extensively explored the logical reasoning capabilities of LLMs, often through benchmarks grounded in symbolic logic. Datasets such as FOLIO, P-FOLIO (an enhanced version of FOLIO featuring artificially constructed sentences), LOGICBENCH, and CONTEXTHUB employ synthetic constructions, e.g., using logical operators like $\wedge$ and $\vee$ to connect propositions, in order to evaluate deductive reasoning. While syntactically rigorous, these formats deviate from natural language and fail to reflect the kinds of reasoning errors encountered in real-world discourse. For example, applying Disjunctive Syllogism in such settings primarily assesses rule-based symbolic manipulation. In contrast, detecting a *False Cause* fallacy, as studied in this work, requires semantic and commonsense reasoning, such as identifying when a correlation is incorrectly interpreted as causation.

Benchmark	NE	CQ	RW	FL
FOLIO (Han et al. 2022)	✓	✗	✗	✗
P-FOLIO (Han et al. 2024)	✓	✗	✗	✗
LogicBench (Parmar et al. 2024)	✓	✗	✗	✗
ContextHub (Hua et al. 2024)	✓	✗	✗	✗
LOGIC (Jin et al. 2022b)	✓	✗	✗	✓
LFUD (Li et al. 2024b)	✓	✗	✗	✓
BIG-Bench (Srivastava et al. 2023)	✓	✗	✗	✓
LogicAsker (Wan et al. 2024)	✓	✗	✗	✓
COIG-CQIA (M-A-P 2024)	✗	✓	✓	✗
RuoZhiBench (Zhai et al. 2025)	✗	✓	✓	✓
SMARTYPAT-BENCH (This work)	✓	✓	✓	✓

Table 1: Comparison of Benchmarks. The abbreviations denote: NE(Native English), CQ(Cunning Question), RW(Real World), FL(Fallacy Label)

LOGICASKER extends the symbolic paradigm with annotations but still lacks coverage of semantically rich fallacies like *False Cause*. More naturalistic efforts such as LOGIC attempt to bridge the gap by sourcing fallacy examples from student exam quizzes. However, these instances are often trivial, overly simplistic, and disconnected from real-world reasoning, thus providing limited challenge to LLMs. Other benchmarks, like LFUD, generate fallacies by prompting LLMs with isolated propositions (e.g., *"Peter visited China last year"*) sourced from Wikipedia or textbooks. Yet these are typically not genuine fallacies and lack coherence or context. Moreover, the fallacy types used in LFUD diverge from those found in natural discourse. Our experiments confirm that generating realistic fallacies poses a much greater challenge for LLMs, as it demands deeper semantic and commonsense reasoning.

The COIG-CQIA benchmark improves realism by sourcing subtle, context-dependent fallacies from Chinese online forums. However, its reliance on direct Chinese–English translations (Zhai et al. 2025) reduces contextual fidelity, and the lack of explicit fallacy labels limits its diagnostic value. Additionally, its fallacy categorization scheme suffers from conceptual ambiguity, with several categories lacking clear definitions or logical coherence (see Section 2).

Table 1 presents a comparative overview of existing logic reasoning benchmarks for LLM evaluation. The limitations discussed above motivate the design of our benchmark.

Construction of SMARTYPAT-BENCH To construct SMARTYPAT-BENCH, we curated logically flawed posts from an English-language subreddit (Reddit 2025) analogous to COIG-CQIA. Using the Arctic Shift dataset (Heitmann 2025), we extracted 251,052 entries and applied keyword filtering, upvote-based selection, and expert annotation, resulting in 502 high-quality examples. Each post was labeled with relevant fallacy types and manually transformed into a logic-constrained form $(p_1 \wedge \cdots \wedge p_n) \rightarrow q$ for analysis. Further details are provided in the technical report.

Abb.	Fallacy Type	Count	Percent (%)
FP	False Premise	218	35.12
EC	Equivocation	189	30.40
FA	False Analogy	88	14.17
NF	Nominal Fallacy	38	6.12
CT	Contextomy	32	5.16
FS	False Cause	11	1.77
AF	Accident Fallacy	8	1.29
ID	Improper Distribution or Addition	7	1.13
BQ	Begging the Question	7	1.13
IE	Inverse Error	6	0.97
WD	Wrong Direction	6	0.97
FD	False Dilemma	5	0.81
FC	Fallacy of Composition	3	0.48
IT	Improper Transposition	3	0.48

Table 2: Fallacy types and their frequencies. The dataset exhibits a substantial class imbalance.

Observations from SMARTYPAT-BENCH Table 2 illustrates the distribution of various fallacy types within SMARTYPAT-BENCH. We observe that *FP*, *EC*, and *FA* are the three most prevalent fallacies, collectively accounting for over 79.7% of the dataset. In contrast, the three least frequent types, *IT*, *FC*, and *FD*, together constitute only 1.77%. This indicates a highly imbalanced representation of fallacy types in user-generated forum posts. Additionally, the entire process of constructing a rigorous benchmark for evaluating logic reasoning from real-world data is labor-intensive and time-consuming. On average, screening each sentence required approximately 0.5 minutes, annotating the logical fallacy types took roughly 3 minutes, and transforming questions into declarative sentences took around 2 minutes. This resulted in a cumulative workload of approximately 3,760 minutes, or roughly 62.67 hours, for a single annotator. Consequently, **to reduce manual effort in developing larger datasets and to maintain complete control over dataset content, there is a need for techniques capable of automatically generating high-quality, logically fallacious statements**.

4 Methodology

The overall workflow of **SMARTYPAT** is formalized in Algorithm 1 and illustrated in the appendix. The method comprises three stages: ❶**PrologProgramDesign** (Section 4): Based on the logical implication format defined in Section 3, we analyze the structural properties of each fallacy type to derive schematic patterns, guiding the design of corresponding *Prolog* predicates. This results in a complete Prolog program capable of supporting systematic knowledge generation (Lines 5–7). ❷**PrologKnowledgeGeneration** (Section 4): Using fallacy-specific predicates from the previous stage, this module invokes an LLM to generate *facts*, which are then injected into the Prolog knowledge base (Lines 8–9). ❸**FallacySentenceTransformation** (Section 4): The enriched Prolog knowledge base is executed to infer fallacy-specific outputs. Only fact combinations satisfying the logical criteria of the target fallacy are retained. These outputs are then transformed into natural language sentences and evaluated to ensure alignment with the intended fallacy type (Lines 10–13).

Algorithm 1: SMARTYPAT

Require: T_{decl}: DeclarativeTemplates, Φ: FallacyTypes, $\mathcal{L}$: LLM, Π: PrologEngine
Ensure: $S_{fallacy}$: GeneratedFallaciousSentences
1: **function** SMARTYPAT($T_{decl}, \Phi, \mathcal{L}, \Pi$)
2: $\mathcal{K}_{prolog} \leftarrow \emptyset$ ▷ Initialize Prolog knowledge base
3: $S_{fallacy} \leftarrow \emptyset$ ▷ Initialize fallacious sentence set
4: **for** $\phi \in \Phi$ **do**
5: $T_\phi \leftarrow$ FILTERBYFALLACYTYPE(T_{decl}, ϕ) ▷ Filter relevant templates by type ϕ
6: $\phi_{syn} \leftarrow$ ANALYZEANDCONSTRUCTSCHEMA(T_ϕ) ▷ Construct synthesized schema
7: $(\tilde{F}_\phi, \tilde{R}_\phi) \leftarrow$ EXTRACTFACTSRULES(ϕ_{syn}) ▷ Extract initial facts and rules from schema
8: $\tilde{F}_{add} \leftarrow$ GENERATEFACTSWITHLLM($\tilde{F}_\phi, \tilde{R}_\phi, \mathcal{L}$) ▷ Expand facts using LLM
9: $\mathcal{K}_{prolog} \leftarrow \mathcal{K}_{prolog} \cup \tilde{F}_{add} \cup \tilde{R}_\phi$
10: Π.assertz($\mathcal{K}_{prolog}$) ▷ Load all facts and rules into Prolog engine
11: $\mathcal{H}_{\tilde{F}_{\text{valid}}} \leftarrow$ *findall*($[\![\tilde{R}_\phi]\!]_{\mathcal{K}_{prolog}}$) ▷ Query Prolog to retrieve valid fact combinations
12: $s_{nl} \leftarrow$ GENERATENLWITHLLM($\mathcal{H}_{\tilde{F}_{\text{valid}}}, T_{phi}, \mathcal{L}$) ▷ Generate NL sentences from valid facts
13: $S_{fallacy} \leftarrow S_{fallacy} \cup \{s_{nl}\}$
14: **end for**
15: **return** $S_{fallacy}$
16: **end function**

Prolog Program Design This step aims to synthesize common reasoning patterns and convert the schemas derived in the previous stage into *Prolog* predicates that support logical verification. The procedure is outlined in Algorithm 1. We begin by leveraging the logical implication format defined for each fallacy type in Section 3 (Line 5). Subsequently, multiple rounds of collaborative analysis were conducted among the co-authors to extract schematic reasoning patterns associated with each fallacy type. For example, the sentence *"Why do meteors always land in craters?"* exemplifies a fallacy that inverts the causal or temporal relationship between observation and explanation—thus allowing a relatively straightforward formalization. In contrast, semantically nuanced fallacies such as *CT* involve distortions of quoted material or partial misinterpretations of intent, as seen in *"If I continue eating an apple a day, will I never get my PhD?"*. These qualitative analyses allow us to formally capture the core structure of each fallacy (Line 6).

We then design corresponding *Prolog* predicates, denoted as `pd`, incorporating both rules and example *facts* to serve as few-shot prompts for the LLM (Line 7). Of the 14 fallacy types in SMARTYPAT-BENCH, 11 were selected for enhancement via this method; the remaining three exhibited sufficiently strong baseline performance (Section 5).

Table 3 summarizes the full set of `pd` predicates and their semantic roles. Each predicate is carefully constructed to encode the specific reasoning flaw of its corresponding fal-

Predicate	Description	Notation
has_effect(α, δ, ϵ)	An action α lasting δ results in ϵ.	HE/3
valid_accumulate(υ, ρ, τ)	Repeating an action of duration υ, satisfying ρ, accumulates to valid result of τ.	VC/3
established_fact(χ, ϕ)	Condition χ establishes fact ϕ.	EF/2
false_premise(ϕ, π)	Fact ϕ has a false premise π.	FP/2
plausible_observation(o, π)	Valid observation o can incorrectly justify π.	PO/2
false_premise_lead_conclusion(π, o, γ)	False premise π along with valid observation o leads to erroneous conclusion γ.	FPLC/3
has_rule(o, ρ)	Object o contains instruction or rule ρ.	HR/2
rule_unreasonable_interpretation(ρ, ι)	Rule ρ could be unreasonably interpreted as ι.	RUI/2
rule_reasonable_interpretation(ρ, ι)	Rule ρ could be reasonably interpreted as ι.	RRI/2
has_property(χ, π)	Component χ has property π.	HP/2
is_part_of(χ, ω)	Component χ is part of whole ω.	IPO/2
lacks_property(ω, π)	Whole ω lacks property π.	LP/2
claim_and_argument(χ, α)	Argument α is used to support claim χ.	CA/2
explicit_meaning_of_argument(α, ϵ)	Argument α explicitly means ϵ.	EMA/2
explicit_meaning_rely_on_claim(ϵ, χ)	Explicit meaning ϵ relies on claim χ.	EMRC/2
quote_context(θ, μ)	Original meaning μ of quote θ.	QC/2
quote_out_of_context(θ, μ)	Quote θ is misinterpreted as μ.	QOC/2
fact_related_out_of_context(μ, ϕ)	Misinterpretation μ is related improperly to fact ϕ.	FROC/2
improper_fact_quote_out_of_context(ϕ, γ)	Fact ϕ improperly leads to conclusion γ.	IFQOC/2
complement_cases(α, β)	Cases α and β complement each other.	CC/2
implies(χ, ρ)	Condition χ logically implies ρ.	IM/2
cause(α, β)	α directly causes β.	CS/2
happen_at(τ, ϵ)	Event ϵ happens in scenario τ.	HA/2
real_cause(χ, ϵ)	Observed effect ϵ is actually caused by χ.	RC/2

Table 3: List of Predicates, Descriptions, and Notations

lacy. Formally, we define $\mathcal{H}_{\tilde{F}_{\text{valid}}}$ as the set of all *Prolog* fact instances satisfying the rule $\tilde{R}_\phi$ for a fallacy type ϕ, with the mapping:

$$\mathcal{H}_{\tilde{F}_{\text{valid}}} \longmapsto \texttt{pd}(\texttt{argument}_1, \texttt{argument}_2, \ldots, \texttt{argument}_n)$$

ensuring that only logically valid constructions under $\tilde{R}_\phi$ are used for natural language generation. Below, we illustrate schemas for 3 example fallacy types; full definitions for all types are provided in the appendix.

Definition 1 Accident Fallacy [AF]. This definition identifies all valid instantiations of (O, R, I, K) that match the rule schema R-AF, which formalizes the accident fallacy, misinterpreting a general rule by extending it beyond its reasonable bounds. Let O=shampoo_bottle, R=lather_rinse_repeat, I=wash_once_or_twice, and K=infinite_washing. Here, $HR(O, R)$ holds since the rule appears on the shampoo bottle. A reasonable interpretation is captured by $RRI(R, I)$: the instruction implies washing once or twice. However, $RUI(R, K)$ also holds: an unreasonable interpretation would suggest one must wash infinitely. Since $I \neq K$, the conclusion formed by treating K as a valid reading commits an accident fallacy. The error arises from rigidly applying a general rule without regard to practical limits or intended scope, leading to an absurd or unintended consequence.

$$\frac{\tilde{R}_\Phi = pd(O, R, I, K) \ :- \ HR(O, R), \quad RRI(R, I), \quad RUI(R, K), \quad I \neq K}{\mathcal{H}_{\tilde{F}_{\text{valid}}} \longmapsto pd(O, R, I, K)} \quad \text{(R-AF)}$$

Definition 2 Improper Distribution or Addition [ID]. This definition identifies all valid tuples (A, Δ, E, Υ) that instantiate the rule. Let X=brush_teeth, A=2_mins, Δ=14_mins, E=teeth_health_for_that_day, Υ=teeth_health_for_one_week, and R=repeat_7_times_in_one_go. The rule schema R-ID captures a reasoning failure where action duration can be validly accumulated, but the corresponding effect cannot. Specifically, $HE(X, A, E)$ holds since brushing for 2 minutes improves dental health for a day, and $HE(X, \Delta, \Upsilon)$ holds since brushing once for 14 minutes yields a week-long effect. Temporal accumulation is valid: $VC(A, R, \Delta)$, as seven repetitions of A compose Δ. However, effect-level accumulation fails: $\neg VC(E, R, \Upsilon)$. Therefore, taking the positive version $VC(E, R, \Upsilon)$ as a premise and instantiating it with such a tuple (A, Δ, E, Υ) leads to a fallacy, as it incorrectly assumes that repeating short-term effects compounds into the long-term effect.

$$\frac{\tilde{R}_\Phi = pd(A, \Delta, E, \Upsilon) \ :- \ HE(X, A, E), \quad HE(X, \Delta, \Upsilon), \quad VC(A, R, \Delta), \quad \neg VC(E, R, \Upsilon)}{\mathcal{H}_{\tilde{F}_{\text{valid}}} \longmapsto pd(A, \Delta, E, \Upsilon)} \quad \text{(R-ID)}$$

Definition 3 False Cause [FS]. This definition identifies all valid instantiations of (T, E) that match the rule schema R-FS, which formalizes the false cause fallacy, wrongly treating the mere temporal or spatial co-occurrence of two events as evidence that one directly produces the other as a substance. Let T=lightbulb_switch, E=darkness_emission, Υ=room_event, and X=absence_of_light. Here, $HA(\Upsilon, T)$ and $HA(\Upsilon, E)$ hold: both the action of switching the lightbulb and the resulting darkness occur as part of the same observable room event. However, $RC(X, E)$ identifies the real cause of darkness as the absence of light, not the switching action itself. Since $X \neq T$

and $T@ < E$ (the switch action precedes the observed effect), the fallacy arises when one concludes that turning off a lightbulb emits darkness as a physical substance. This misrepresents a lack (the absence of illumination) as a generative act, conflating temporal correlation with causal production.

$$\frac{\tilde{R}_{\Phi} = pd(T,E) \ :- \begin{array}{ll} HA(\Upsilon,T), & HA(\Upsilon,E), \\ RC(X,E), & X \neq T, \quad T@ < E \end{array}}{\mathcal{H}_{\tilde{F}_{\text{valid}}} \longmapsto pd(T,E)} \tag{R-FS}$$

Prolog Knowledge Generation To address the challenge of automatically generating statements containing nuanced logical fallacies, we leverage the facts and rules constructed in the previous section, combined with LLMs, to reduce human effort. Specifically, we provide the LLM with the formal rule defining the fallacy type and corresponding *Prolog* facts as few-shot examples (Line 8). An important observation from our experiments is that LLMs better capture inter-predicate relationships when facts related to a specific fallacy instance are grouped together rather than by predicate name. For instance, in Equation R-AF, it is more effective to group *HR(O, R)*, *RRI(R, I)*, and *RUI(R, K)* within a single example. This grouping encourages the LLM to semantically align argument values and generate a logically coherent combination of *HR*, *RRI*, and *RUI* predicates.

Moreover, because predicates encode relationships between arguments, adding inline comments to clarify each predicate improves the LLM's understanding. For example, the fact `HR(highway, maximum_speed_65)` can be annotated as `% this means highway has a rule of maximum speed 65`. These comments help LLMs more accurately infer the semantics of each predicate. The prompt used for this task is presented in the appendix.

Fallacy Sentence Transformation This section aims to transform appropriate fact combinations into natural language sentences that reflect the implication-style format established in Section 3. Specifically, we utilize the knowledge base $\mathcal{K}_{\text{prolog}}$ (Lines 9–10) and construct a query to extract all valid *Prolog* knowledge facts corresponding to the rules of a given fallacy type, denoted as $\mathcal{H}_{\tilde{F}_{\text{valid}}}$ (Line 11). Finally, we instruct the LLM using both the template set T_{ϕ} and the retrieved facts $\mathcal{H}_{\tilde{F}_{\text{valid}}}$ to convert the logical facts into natural language sentences (Line 12). The prompt used for sentence transformation is shown in the appendix. We denote the resulting dataset as **SMARTYPAT-BENCH-AUGMENTED**.

5 Experimentation

SMARTYPAT employs a *Prolog*-based backend comprising 1,458 lines of code, including 24 unique fact predicates, 13 rule predicates, and 11 distinct queries. In addition, we implement 12 Python scripts (1,517 lines total) for managing LLM interaction, data preprocessing, metric computation, and visualization. Fact generation is powered by Claude 3.7 with extended thinking mode (Anthropic 2025b).

Our experiments are designed to answer the following research questions:

- **RQ1 (Fallacy Generation Quality):** How can we construct a logical fallacy generator that reliably produces sentences reflecting specific fallacy types?
- **RQ2 (LLM Fallacy Detection Capability):** To what extent can LLMs detect the presence of a logical fallacy in a given sentence?
- **RQ3 (LLM Fallacy Categorization Capability):** Can LLMs correctly categorize a fallacious sentence using the appropriate fallacy labels?

Experiment Setup

Baseline Methods We compare SMARTYPAT with two baseline methods that omit predicate engineering and structured logic definitions, relying solely on LLM internal reasoning. Details follow:

- **FallacyGen-Direct:** This baseline uses Claude 3.7 (Anthropic 2025b), a state-of-the-art reasoning model outperforming Deepseek-R1 (DeepSeek 2025a) and GPT-o3-mini (OpenAI 2025c) on benchmarks such as GPQA Diamond (Anthropic 2023). Following a simple pipeline, we (1) compile example sentences and formal definitions for each fallacy, (2) prompt the LLM with both, and (3) instruct it to generate new instances of the same fallacy type. Variants of the prompt were explored but yielded similar results. Full prompts are detailed in the appendix.
- **FallacyGen-Prolog:** This method prompts LLMs to generate *Prolog* programs, including facts, predicates, and inference rules, based on each fallacy's definition and illustrative examples. The LLM is also tasked with generating corresponding natural language sentences. Unlike SMARTYPAT, this approach imposes no structural constraints or validation mechanisms, allowing the model full control over rule construction. Details are provided in the appendix.

Benchmarks and Tools We summarize the datasets used for each research question and the tools involved in our experiments:

- **SMARTYPAT-BENCH:** Used across RQ1, RQ2, and RQ3. For RQ1, it serves as the basis for validating sentence quality evaluation. For RQ2 and RQ3, it is employed to assess LLM capabilities in logical fallacy detection and categorization.
- **SMARTYPAT-BENCH-AUGMENTED:** Also used in RQ1–RQ3. In RQ1, we reference SMARTYPAT-BENCH-AUGMENTED as the output of SMARTYPAT, enabling direct method comparison. In RQ2 and RQ3, it complements SMARTYPAT-BENCH to evaluate LLM performance across both synthetic and real-world data.
- **BENIGN-BENCH:** Used in RQ2 and RQ3. We compile logically sound sentences from **C4** and **FineWeb** (for AI 2019; FineWeb 2024), filtering for sentence tokens to match `r/ShittyAskScience`, yielding a curated set of 502 examples. This benchmark is combined with SMARTYPAT-BENCH and SMARTYPAT-BENCH-AUGMENTED to compute metrics such as false positive (FPR) rates.
- **Prolog Infrastructure:** We use **SWI-Prolog** (Wielemaker et al. 2024), a robust, open-source implementation of the Prolog language widely used in logic programming.

Evaluated Models We evaluate nine state-of-the-art LLMs chosen for their relevance to logical reasoning. These models fall into two categories: **reasoning models**, which produce intermediate logic chains (e.g., DeepSeek R1 (DeepSeek 2025a), GPT-o3-mini-2025-01-31 (OpenAI 2025b), Claude 3.7 with extended thinking (Anthropic 2025b)); and **non-reasoning models**, which do not explicitly perform stepwise reasoning (e.g., Claude 3.5 (Anthropic 2025a), LLaMA 3.1 405B (AI 2025), DeepSeek V3 (DeepSeek 2025b), GPT-4o-2024-08-06 (OpenAI 2025a), Grok-2 (xAI 2025), Claude 3.7-20250219 (Anthropic 2025b)). They are accessed via Cloud APIs. Full prompting templates are provided in the appendix.

RQ1: Fallacy Generation Quality

To systematically evaluate fallacy generation quality, we compare three methods: **FallacyGen-Direct**, **FallacyGen-Prolog**, and **SMARTYPAT**. For each selected fallacy type, each method generates 20 unique sentences, producing a balanced dataset for per-fallacy comparison.

Sentence Quality Evaluator To reduce evaluation bias, we adopt a cross-model evaluation strategy. All generations are produced using **Claude 3.7 with extended thinking**, while evaluation is performed by **GPT-4o**. This separation ensures that evaluation reflects general sentence quality, independent of the generation model. Each sentence is scored on a 0–3 scale, with 3 indicating strong alignment with the intended fallacy. All evaluations are run at temperature 0, and each sentence is scored three times. Evaluation prompt details are provided in the appendix.

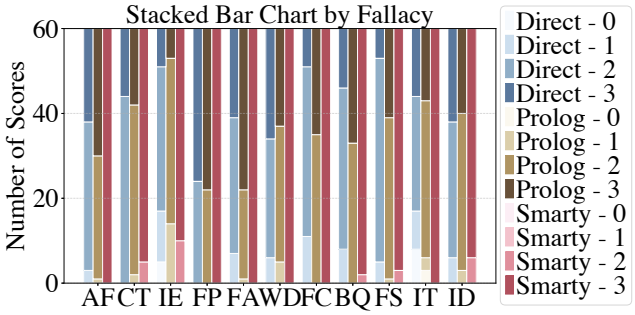

Figure 1: The score distribution of the three methods across different types of logical fallacies. *Direct means **FallacyGen-Direct**, Prolog means **FallacyGen-Direct**.More score 3 is better.

Sentence Quality Evaluator Validity. We first validated the evaluator's effectiveness on the original SMARTYPAT-BENCH, and further confirmed its consistency with human annotations. Details of the prompts, annotation process, and agreement statistics are provided in the Appendix.

Baseline Testing and Analysis. FallacyGen-Direct excels at generating fallacies reliant on surface-level features, such as *EC*, *NF*, and *FD*. These fallacies often rely on shallow lexical ambiguity or binary framing, patterns LLMs can easily mimic without deep reasoning. For example, *EC* and *NF* exploit ambiguity in terms like *pound* or phrases like *burn calories*, while *FD* leverages binary constructions. This highlights that LLMs perform well on linguistically superficial fallacies but struggle with semantically complex ones. In contrast, fallacies like *CT* require context or sociocultural awareness, posing greater difficulty. We therefore exclude these trivial fallacies from Prolog-based generation, as FallacyGen-Direct already performs near optimally on them.

SMARTYPAT Effectiveness. As shown in Figure 1, SMARTYPAT significantly reduces low-quality outputs (scores 0–1) and increases high-quality outputs (scores 2–3) across nearly all fallacy types. Notably, for *FC*, SMARTYPAT generates 60 score-3 instances, outperforming FallacyGen-Prolog (25) and FallacyGen-Direct (9), demonstrating superior semantic and structural alignment. Additionally, semantic similarity analysis using `text-embedding-3-large` (OpenAI 2024) shows intra- and inter-benchmark cosine similarity of 0.16, confirming the novelty of generated content. Full results and embeddings are detailed in the appendix.

Comparison of Average Scores. We report average sentence quality scores by fallacy type across methods; the full table is provided in the appendix. SMARTYPAT consistently achieves the highest average score in all categories. The "Enhance" row quantifies the relative improvement of SMARTYPAT over FallacyGen-Direct. On average, SMARTYPAT outperforms this baseline by 38.12%, with particularly large gains observed in ***IE (+58.88%)***, ***IT (+62.16%)***, ***FC (+52.54%)***, *and* ***FS (+45.08%)***. These results highlight SMARTYPAT's advantage in enhancing both structural accuracy and semantic fidelity, demonstrating its effectiveness for generating high-quality, logic-driven fallacy instances.

RQ2: LLM Fallacy Existence Detection Capability

We evaluate nine state-of-the-art LLMs on their ability to detect the presence of logical fallacies in sentences. Our analysis addresses two key dimensions: overall detection performance and fallacy-specific detection difficulty.

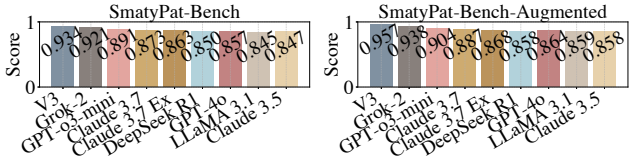

Figure 2: F1 score(higher better), sorted by F1 score in descending order. Claude 3.7 Ex means Claude 3.7 Extended Thinking.

Logical Fallacy Detection Ability We report *FPR*, where logically sound sentences are misclassified as fallacious; *FNR*, where fallacies are missed; and the resulting F1-score. Figure 2 presents the F1 scores, with the complete figure available in the appendix. SMARTYPAT-BENCH-AUGMENTED exhibits FPR/FNR/F1 patterns comparable to SMARTYPAT-BENCH, indicating LLMs perceive both datasets as similarly fallacious. Interestingly, non-reasoning models (e.g., DeepSeek V3, Grok-2) consistently outperform reasoning models (e.g., Claude 3.7, GPT-o3-mini) in F1-score, despite expectations to the contrary. Closer inspection suggests reasoning models tend to overanalyze, flagging benign content

as fallacious. For instance, Claude 3.7 mislabels a simple instructional sentence ('If you are a beginner, it is best to begin with a flat board') as an *AF* due to overgeneralization. This behavior likely reflects a form of *confirmation bias* (O'Leary 2025), where models assume fallacies must be present. All models show high FPR and near-zero FNR, with reasoning models generally more prone to false positives. **These findings reveal a sensitivity bias in LLMs, with reasoning models not outperforming non-reasoning models in detection accuracy.**

Figure 3: LLM accuracy (darker is better) in identifying fallacies from SMARTYPAT-BENCH (without *) and SMARTYPAT-BENCH-AUGMENTED (with *).

Fallacy-wise Detection Ability. Using SMARTYPAT-BENCH-AUGMENTED, we compute fallacy-wise detection accuracy (Figure 3) by checking whether the ground-truth fallacy label appears in the model's predictions. Fallacy types in SMARTYPAT-BENCH-AUGMENTED are notably easier to identify. Averaged across models, LLMs perform best on *FS* and *FA*, consistent with intuitive causal and analogical reasoning. In contrast, detection rates are lower for context-dependent fallacies such as *CT*, *IT*, and *ID*. **These results suggest that LLMs handle surface-level causal or analogical fallacies well but struggle with those requiring nuanced contextual understanding.**

RQ3: LLM Fallacy Categorization Capability

This section evaluates whether LLMs can accurately assign fallacy labels to given sentences. We test nine LLMs on both SMARTYPAT-BENCH and SMARTYPAT-BENCH-AUGMENTED. Unlike the detection task (Figure 3), this task assesses the overall similarity between predicted and ground-truth label sets, factoring in prediction rank and the presence of incorrect labels.

Ranked Fallacy Scorer. We adopt a rank-weighted scoring function. Let $G = [g_1, g_2, \ldots, g_m]$ be the ground-truth labels and $P = [p_1, p_2, \ldots, p_n]$ the predicted labels. The score is calculated by:

$$S_{\text{original}}(G, P) = \sum_{i=1}^{n} \begin{cases} \frac{1}{i}, & \text{if } p_i \in G \\ -\frac{1}{i}, & \text{if } p_i \notin G \end{cases}$$

For instance, a correct top-ranked prediction yields $+1$, while an incorrect one yields -1. The worst-case score occurs when none of the predicted labels appear in G, particularly when predicting nearly all possible labels ($T - 1$, where T is the total number of fallacy types), leading to a penalty of $-\sum_{i=1}^{T-1} \frac{1}{i}$.

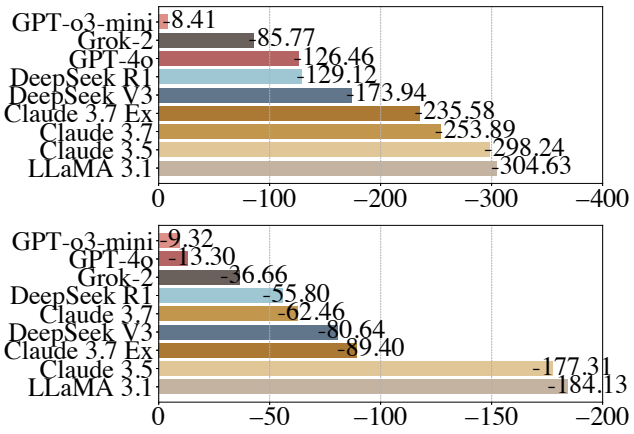

Figure 4: Fallacy label scores for selected LLMs, sorted in descending order (Close to the Left is better). The top is SmartyPat-Bench.

Fallacy Categorization Capabilities. Figure 4 shows model scores on SMARTYPAT-BENCH (top) and SMARTYPAT-BENCH-AUGMENTED (bottom). Scores on SMARTYPAT-BENCH-AUGMENTED are consistently higher, reflecting more explicit fallacies structures. Claude 3.7 and its extended version improve by over 70%, likely due to alignment with familiar generation patterns (Panickssery, Bowman, and Feng 2024). Reasoning models generally outperform non-reasoning ones; notably, GPT-o3-mini surpasses GPT-4o, and DeepSeek R1 exceeds DeepSeek V3. **Overall, reasoning-oriented models are better suited for fallacy classification.**

We also evaluate Grok-2's behavior (see Appendix for details, limitations, and future work). Its unexpectedly strong performance arises from a conservative labeling strategy that minimizes penalties. Moreover, the sentences generated by SMARTYPAT exhibit clearer and more explicit fallacy structures, illustrating how the integration of LLMs with formal methods can enhance output stability and enable verifiable generation, which is an emerging direction toward the trustworthy deployment of LLMs in critical applications.

6 Conclusion

We introduce SMARTYPAT-BENCH, the first real-world native English dataset of fallacious questions, and SMARTYPAT, a Prolog-based generation method that produces SMARTYPAT-BENCH-AUGMENTED. SMARTYPAT outperforms two baselines in accuracy and diversity. Evaluation of nine state-of-the-art LLMs shows that stronger models often overanalyze, leading to false positives: non-reasoning models excel at detection, while reasoning models perform better at categorization. The GPT family, particularly GPT-o3-mini, achieves the best overall balance.

References

AI, M. 2025. LLaMA 3.1 405B. Available at https://ai.meta.com/llama/.

Anthropic. 2023. Claude 3.7: Sonnet. Accessed: 2025-03-21.

Anthropic. 2025a. Claude 3.5. Available at https://www.anthropic.com.

Anthropic. 2025b. Claude 3.7. Available at https://www.anthropic.com.

DeepSeek. 2025a. DeepSeek R1. Available at https://www.deepseek.com.

DeepSeek. 2025b. DeepSeek V3. Available at https://www.deepseek.com.

FineWeb, H. 2024. FineWeb: High-Quality Web Text Data for LLM Training. Accessed: March 3, 2025.

for AI, A. I. 2019. C4: Colossal Clean Crawled Corpus. Accessed: March 3, 2025.

Han, S.; Schoelkopf, H.; Zhao, Y.; Qi, Z.; Riddell, M.; Zhou, W.; Coady, J.; Peng, D.; Qiao, Y.; Benson, L.; et al. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Han, S.; Yu, A.; Shen, R.; Qi, Z.; Riddell, M.; Zhou, W.; Qiao, Y.; Zhao, Y.; Yavuz, S.; Liu, Y.; et al. 2024. P-FOLIO: Evaluating and improving logical reasoning with abundant human-written reasoning chains. *arXiv preprint arXiv:2410.09207*.

Heitmann, A. 2025. Arctic Shift: Making Reddit Data Accessible. Accessed: March 3, 2025.

Hua, W.; Zhu, K.; Li, L.; Fan, L.; Lin, S.; Jin, M.; Xue, H.; Li, Z.; Wang, J.; and Zhang, Y. 2024. Disentangling logic: The role of context in large language model reasoning capabilities. *arXiv preprint arXiv:2406.02787*.

Humbatova, N.; Jahangirova, G.; and Tonella, P. 2021. DeepCrime: mutation testing of deep learning systems based on real faults. *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*.

Jin, Z.; Lalwani, A.; Vaidhya, T.; Shen, X.; Ding, Y.; Lyu, Z.; Sachan, M.; Mihalcea, R.; and Schoelkopf, B. 2022a. Logical Fallacy Detection. In Goldberg, Y.; Kozareva, Z.; and Zhang, Y., eds., *Findings of the Association for Computational Linguistics: EMNLP 2022*, 7180–7198. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

Jin, Z.; Lalwani, A.; Vaidhya, T.; Shen, X.; Ding, Y.; Lyu, Z.; Sachan, M.; Mihalcea, R.; and Schoelkopf, B. 2022b. Logical fallacy detection. *arXiv preprint arXiv:2202.13758*.

Li, N.; Li, Y.; Liu, Y.; Shi, L.; Wang, K.; and Wang, H. 2024a. Drowzee: Metamorphic Testing for Fact-Conflicting Hallucination Detection in Large Language Models. *Proc. ACM Program. Lang.*, 8(OOPSLA2).

Li, Y.; Wang, D.; Liang, J.; Jiang, G.; He, Q.; Xiao, Y.; and Yang, D. 2024b. Reason from fallacy: Enhancing large language models' logical reasoning through logical fallacy understanding. *arXiv preprint arXiv:2404.04293*.

Li, Y.; Zhou, Q.; Luo, Y.; Ma, S.; Li, Y.; Zheng, H.-T.; Hu, X.; and Yu, P. S. 2025. When LLMs meet cunning texts: A fallacy understanding benchmark for large language models. *Advances in Neural Information Processing Systems*, 37: 112433–112458.

M-A-P. 2024. COIG-CQIA (Ruozhiba) Dataset. https://huggingface.co/datasets/m-a-p/COIG-CQIA. Accessed: March 2025.

Ma, L.; Juefei-Xu, F.; Zhang, F.; Sun, J.; Xue, M.; Li, B.; Chen, C.; Su, T.; Li, L.; Liu, Y.; Zhao, J.; and Wang, Y. 2018. DeepGauge: multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE '18, 120–131. New York, NY, USA: Association for Computing Machinery. ISBN 9781450359375.

OpenAI. 2024. New embedding models and API updates. Accessed: 2025-07-22.

OpenAI. 2025a. GPT-4o. Available at https://openai.com/research/gpt-4o.

OpenAI. 2025b. O3 Mini. Available at https://openai.com.

OpenAI. 2025c. OpenAI o3-mini. Accessed: 2025-03-20.

O'Leary, D. E. 2025. Confirmation and Specificity Biases in Large Language Models: An Explorative Study. *IEEE Intelligent Systems*, 40(1): 63–68.

Panickssery, A.; Bowman, S. R.; and Feng, S. 2024. LLM Evaluators Recognize and Favor Their Own Generations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Parmar, M.; Patel, N.; Varshney, N.; Nakamura, M.; Luo, M.; Mashetty, S.; Mitra, A.; and Baral, C. 2024. LogicBench: Towards systematic evaluation of logical reasoning ability of large language models. *arXiv preprint arXiv:2404.15522*.

Reddit. 2025. r/shittyaskscience - The Place for Bad Science Questions. Accessed: 2025-03-01.

Srivastava, A.; Rastogi, A.; Rao, A.; Shoeb, A. A.; Abid, A.; Fisch, A.; Brown, A. R.; Santoro, A.; Gupta, A.; Garriga-Alonso, A.; et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*.

Wan, Y.; Wang, W.; Yang, Y.; Yuan, Y.; Huang, J.-t.; He, P.; Jiao, W.; and Lyu, M. R. 2024. LogicAsker: Evaluating and improving the logical reasoning ability of large language models. *arXiv preprint arXiv:2401.00757*.

Wielemaker, J.; Anjewierden, A.; Schrijvers, T.; et al. 2024. SWI-Prolog – A Comprehensive Prolog Environment. https://www.swi-prolog.org/. Accessed: 2025-03-24.

xAI. 2025. Grok-2. Available at https://x.ai.

Xie, X.; Ma, L.; Juefei-Xu, F.; Xue, M.; Chen, H.; Liu, Y.; Zhao, J.; Li, B.; Yin, J.; and See, S. 2019. DeepHunter: a coverage-guided fuzz testing framework for deep neural networks. *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*.

Xu, Z.; Ding, J.; Lou, Y.; Zhang, K.; Gong, D.; and Li, Y. 2025. Socrates or Smartypants: Testing Logic Reasoning Capabilities of Large Language Models with Logic Programming-based Test Oracles. *arXiv preprint arXiv:2504.12312*.

Zhai, Z.; Li, H.; Han, X.; Zhang, Z.; Zhang, Y.; Baldwin, T.; and Li, H. 2025. RuozhiBench: Evaluating LLMs with Logical Fallacies and Misleading Premises. *arXiv preprint arXiv:2502.13125*.