

LogicSkills: A Structured Benchmark for Formal Reasoning in Large Language Models

Brian Rabern^{1,2} and Philipp Mondorf^{3,4} and Barbara Plank^{3,4}

¹Niche, Inc., Bend, OR, USA

²Oregon State University – Cascades, Bend, OR, USA

³MaiNLP, LMU Munich, Germany

⁴Munich Center for Machine Learning (MCML), Munich, Germany

brian.rabern@gmail.com

p.mondorf@lmu.de

b.plank@lmu.de

Abstract

Large language models perform well on many logical reasoning benchmarks, but it remains unclear which core logical skills they truly master. To address this, we introduce LOGICSKILLS, a benchmark that isolates three fundamental logical skills: (i) *formal symbolization*—translating premises into first-order logic; (ii) *countermodel construction*—showing that an argument is logically invalid by constructing a finite countermodel; and (iii) *validity assessment*—determining whether a conclusion follows from a set of premises. Items are drawn from the two-variable fragment of first-order logic without identity and are presented in both English and a Carrollian nonce-word language. All instances are solver-verified with Z3 for correctness and non-triviality. Across conventional instruction-tuned LLMs, performance is high on *validity assessment* but substantially lower on *formal symbolization* and *countermodel construction*, highlighting that high task-level accuracy can mask weaknesses in core logical skills. In contrast, recent reasoning-tuned models perform strongly across all three tasks, suggesting a more systematic logical skill profile.

1 Introduction

Reasoning in large language models (LLMs) is often evaluated *behaviorally*—by whether systems succeed on tasks that humans would solve through explicit reasoning—without presupposing any particular internal method (Huang and Chang, 2023; Mondorf and Plank, 2024a; Chen et al., 2026). In this setting, logical problems serve as external *proxies* for deductive competence: success should reflect sensitivity to logical structure rather than reliance on superficial cues or world knowledge. However, common benchmarks of logical competence (Yang et al., 2023) often conflate distinct logical subskills, making it difficult to determine which skills LLMs genuinely master.

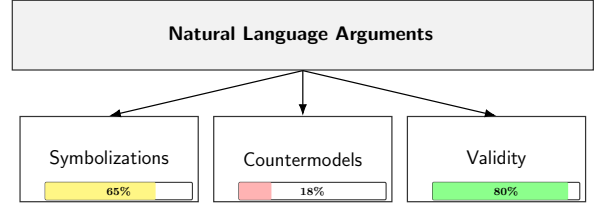


Figure 1: Accuracy of standard instruction-tuned LLMs.

For instance, a model might correctly carry out a formal proof when the premises are already in symbolic form, yet fail to translate natural-language sentences into that form. Likewise, a model might apply modus ponens and other inference rules reliably yet falter when asked to construct a countermodel demonstrating that an argument is invalid. Proof-theoretic facility of this kind leaves model-theoretic understanding untested. Existing benchmarks such as PROOFWRITER (Tafjord et al., 2021) and LOGICBENCH (Parmar et al., 2024) evaluate inference over templated natural language, bundling parsing and inference into a single composite score rather than isolating either.

To address this issue, we introduce LOGICSKILLS, a benchmark for formal reasoning that disentangles fundamental logical skills and exposes failure modes that composite tasks may hide. Drawing on formal logic pedagogy, LOGICSKILLS isolates three core logical skills: (i) *formal symbolization* (mapping sentences to logical form), (ii) *countermodel construction* (demonstrating invalidity via model-theoretic falsification), and (iii) *validity assessment* (judging whether a conclusion follows from a set of premises). These skills are closely related: assessing validity typically requires mapping sentences to logical form and checking whether a countermodel exists in which the premises are true and the conclusion is false. LOGICSKILLS is solver-verified (via Z3 (de Moura and Bjørner, 2008)) and includes a bilingual task design (English and Carrollian), with the Carrollian nonce-word

condition isolating reasoning from prior semantic knowledge.

Systematic evaluations on LOGICSKILLS across frontier open-weight and closed-weight LLMs reveal a notable pattern: conventional instruction-tuned LLMs perform strongly on *validity assessment* but show substantial weaknesses in *formal symbolization* and *countermodel construction* (see Figure 1). This dissociation is notable, since a competent validity judgment must involve some representation of logical structure and some form of model-theoretic check, suggesting that such models may reach correct validity judgments without robustly symbolizing the premises or explicitly constructing countermodels. By contrast, reasoning-tuned LLMs perform strongly across all three skills, indicating a shift toward a more systematic logical skill profile. Together, our results call for more nuanced evaluations of logical reasoning in LLMs.

2 Related Work

Prior evaluations of logical reasoning in LLMs divide broadly into two traditions. *Puzzle-based benchmarks* (e.g., knights-and-knaves, logic-grid, and minesweeper tasks) probe combinatorial reasoning often revealing brittleness under search or minor perturbations (Mondorf and Plank, 2024b; Lin et al., 2025; Giadikiaroglou et al., 2024; Li et al., 2024). *Rule- or proof-oriented suites* such as PROOFWRITER, LOGICBENCH, and LOGICASKEER assess inference templates across propositional and first-order logic (Tafjord et al., 2021; Parmar et al., 2024; Han et al., 2024; Wan et al., 2024).

LOGICSKILLS complements both traditions by offering a structured, skill-level evaluation framework rather than a single composite reasoning task.

3 Benchmark Overview

We build a compositional generator that produces grammatical natural-language sentences paired with formulas in the two-variable fragment of first-order logic (FO^2 , without identity). We choose FO^2 since this logic retains decidability and supports finite satisfiability while being expressive enough to capture nontrivial reasoning (Scott, 1962; Mortimer, 1975). Every formula is realized twice: (i) in controlled English and (ii) in a Carroll-style nonsense lexicon (Carroll, 1871). The latter provides a structurally identical surface form that removes real-world associations and isolates pure logical

competence. For example, the sentences “A human chased a donkey that chased it” and “A borogove snicker-snacked a tove that snicker-snacked it” both map to the formula:

$$\exists x \exists y ((Nx \wedge My) \wedge (Rxy \wedge Ryx)) \quad (1)$$

By varying quantifier structure, predicate choice, name assignments, connectives, and recursive clausal patterns, the generator can produce a wide range of sentences in both English and Carrollian.

Because every sentence is paired with an FO^2 formula, we leverage SMT solvers, whose search procedures descend from Davis and Putnam (1960)—in particular the state-of-the-art theorem prover Z3 (de Moura and Bjørner, 2008)—to compute core semantic properties: (i) whether two formulas are equivalent, (ii) whether a set of formulas is jointly satisfiable, (iii) whether one formula follows from others, and (iv) whether a proposed finite model satisfies (or falsifies) a given argument. This enables large-scale, correctness-guaranteed construction and verification of all deductive instances in the benchmark.

To construct arguments, we sample sets of 3–5 jointly satisfiable premises and use Z3 to search for semantically relevant, non-trivial conclusions that logically follow (valid). For each valid conclusion, we additionally generate 5 structurally similar distractor conclusions that do not follow (invalid). Appendix B provides further details about the dataset and problem construction pipeline.

3.1 Task types

We instantiate three task types, each aligned with a canonical logical skill (Kalish and Montague, 1964). Exact prompts can be found in Appendix F.

Formal symbolization Given a sentence in English/Carroll and a fixed symbol key, models must output a single well-formed formula capturing the sentence using only the provided predicates, constants, and standard logical operators (i.e., $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$, $\exists$).

Input:

Sentence: “All raths are uffish.”
Key: F : “ x is a rath”, G : “ x is uffish”.

Output:

$\forall x (Fx \rightarrow Gx)$

Countermodel construction For an invalid argument, models must provide a finite structure (on

a fixed small domain) that makes all premises true and the conclusion false.¹

Input:
Invalid Argument: $\forall x (Fx \rightarrow Gx), Ga \models Fa$.
Domain: $\{0, 1\}$

Output: $F = \{0\}, G = \{0, 1\}, a = 1$.

Validity assessment Given premises and six candidate conclusions in English/Carroll, models must decide which conclusion(s), if any, *must* follow.

Input:
Premises: “All raths are uffish”; “Zindle is a rath.”
Candidate Conclusions:
1. Zindle isn’t uffish.
2. Only raths are uffish.
3. Zindle is uffish.
4. No raths are uffish.
5. Zindle snicker-snacked a rath.
6. Some rath is not uffish.

Output: 3

4 Evaluation Setup

The compositional data generator described in Section 3 is fully programmatic and supports the creation of a large number of unique problem instances for each task type. For example, we generate over 200k problem instances for the PEFT experiments in Section 5.3. For the main evaluation, however, we use a fixed set of 1,500 problem instances to keep evaluation costs manageable while ensuring broad coverage across tasks. Specifically, our evaluation set comprises 600 problems from *formal symbolization*, 600 from *validity assessment*, and 300 from *countermodel construction*. Symbolization and validity are evaluated bilingually (300 problems in English and 300 in Carroll), whereas countermodel construction is language-neutral. Code for dataset generation, task construction, and model evaluation is publicly available at <https://github.com/brianrabern/LogicSkills>.

We evaluate a range of open- and closed-weight LLMs: *Llama-3.1-8B*, *Llama-3.1-70B*, *Qwen2.5-Math-72B*, *Claude-3.7-Sonnet*, *GPT-4o*, *Gemini-2.5-Flash*, *DeepSeek-Chat*, and *Phi-4*. We further assess two reasoning-tuned models: *Qwen3-32B* and *OpenAI-o3*. All models are evaluated with greedy decoding. The validity and countermodel

prompts permit any solution strategy and place no constraint on intermediate reasoning; only the symbolization prompt restricts the response to a single formula, to ensure parseability (Appendix F). Responses are processed in two stages. First, an extractor LLM (GPT-4o) cleans and normalizes raw completions.² Second, task-specific verification procedures check correctness.

Symbolization Normalized formulas are first checked for an exact match; if necessary, they are parsed and repaired. Cosmetic variants (e.g., $\wedge$ as &, $\neg$ as $\sim$, $\rightarrow$ as $\supset$, or $R(x, y)$ vs. Rxy) are handled via the extractor LLM. Final parses are validated for logical equivalence using Z3.

Countermodels Candidate models are normalized into domain, constants, and predicate extensions. The system enforces *symbol completeness* and validates type/domain consistency (integer domains, correct arities). Candidate models are then translated into SMT-LIB, merged with the negated argument, and checked for satisfiability with Z3.

Validity Extracted index sets are compared against the gold sets under set equality (see Appendix C).

Notably, the evaluation for symbolization and countermodels is semantic (via Z3) rather than string-based: symbolizations are judged by logical equivalence, and countermodels by whether they satisfy the premises and falsify the conclusion.

5 Results

Figure 2 illustrates model accuracy across the three task types described in Section 3. A clear pattern emerges among instruction-tuned models: performance on *validity assessment* is high, whereas performance on *formal symbolization* and especially *countermodel construction* remains substantially weaker. For example, GPT-4o achieves 86.3% on *validity assessment* but only 10.0% on *countermodel construction*; Gemini-2.5-Flash reaches 95.2% versus 13.3%, respectively. While accuracy on *validity assessment* clusters near ceiling, suggesting that premise–conclusion classification is largely mastered under the given conditions, *formal symbolization* remains a consistent bottleneck, reflecting persistent difficulty in mapping natural-language statements to formal representations. The

¹Every item is verified to admit a countermodel for the given domain; see Appendix B.

²A meta-evaluation shows 98–99% extraction accuracy; see Appendix E for more details.

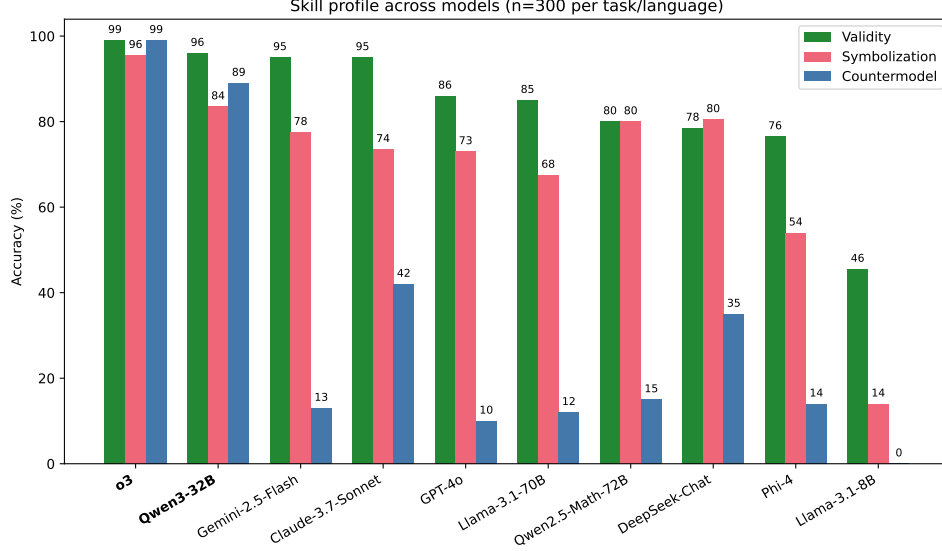


Figure 2: **Skill profile across models** (accuracy %; $n=300$ per task/language). Grouped bars show *Validity*, *Symbolization*, and *Countermodel*. Most models score highly on validity but lag on subskills, especially countermodels.

countermodel construction task is the most discriminative, revealing substantial gaps in model-theoretic reasoning, with most conventional LLMs failing to generate consistent falsifying structures. Taken together, these results suggest deficits in fundamental logical subskills that prior high-level composite tasks may have obscured.

In contrast, reasoning-tuned models such as *Qwen3-32B* and *o3* exhibit strong performance across all three task types. For instance, *o3* achieves 98.8% on *validity assessment*, 98.7% on *countermodel construction*, and 95.3% on *formal symbolization*. This pattern suggests that reasoning-tuned models may be moving toward more comprehensive logical competence.

5.1 Bilingual evaluation

Perhaps surprisingly, bilingual evaluation shows only minor differences: averaging across models, English and Carroll results are closely aligned (validity 83.0% vs. 84.4%; symbolization 70.7% vs. 69.2%), with a maximum difference of 6.7 percentage points across all conditions. This null result suggests that the models are not relying primarily on semantic heuristics when processing the syntactic structure of the arguments. For an overview of accuracy values per model, task, and language, we refer to Table 2 in the Appendix.

5.2 Error analysis

Error analyses across tasks reveal distinct but converging weaknesses in model reasoning. In the *for-*

mal symbolization task, over half of the 1,803 incorrect outputs (50.2%) are semantically incorrect despite syntactic well-formedness, while the remainder show structural issues such as predicate-arity mismatches or ill-formed parentheses. In the *countermodel construction* task, most erroneous outputs (72.9% of 2,020) are structurally well-formed but fail to falsify the argument—typically because one or more premises are false—while the rest (27.1%) reflect missing interpretations or malformed predicate definitions. In the *validity assessment* task, a majority of the 980 incorrect outputs (58.4%) include the correct answer alongside additional invalid options, suggesting difficulty in eliminating plausible distractors. Further details about the error analysis procedure can be found in Appendix D.

5.3 Subskill Transfer via PEFT

In principle, skill at *formal symbolization* and *countermodel construction* should reinforce *validity assessment*: to judge validity, a model must map sentences to their logical form and evaluate whether any counterexample structure can satisfy the premises while falsifying the conclusion. To assess potential cross-skill transfer, we conduct a fine-tuning study. We fine-tune *Llama-3.2-3B-Instruct* with LoRA adapters on 100k *symbolization* and 100k *countermodel* examples (targets generated via tpg; Schwarz, 2025), using a single epoch with AdamW and standard regularization; evaluation items were held out of the training data, and the fine-tuned models are evaluated on the English por-

tion of each task ($n=300$).

We observe large *in-task* gains—symbolization rises from 8.3% to 95.0% and countermodel construction from 0.0% to 23.0%—but no transfer to *validity assessment* (Table 1). Validity accuracy declines under joint training (−15.3%), but this is not a decline in judgment *per se*: fine-tuning sharply reduces the proportion of responses that resolve into a selection among the candidate conclusions (97.3% for the base model, 61.0% under joint training),³ while accuracy among responses that do resolve is essentially unchanged (37.7% base vs. 35.0% joint). Fine-tuned models carry the response format of the training tasks into the validity prompt, producing structured derivations that never close on an answer. The subskills are thus acquired in isolation, without composing into a workflow the model can deploy on the validity task. The pattern parallels a student trained on translation and model construction but never shown how those skills bear on assessing an argument: the components are present, but nothing connects them to the task.

Train Data	Sym.		Counter		Val.	
	Acc.	Δ	Acc.	Δ	Acc.	Δ
Base	8.3	—	0.0	—	36.7	—
Symb. (100k)	95.0	+86.7	0.0	+0.0	39.7	+3.0
Counter (100k)	4.3	−4.0	23.0	+23.0	32.0	−4.7
Joint (200k)	95.7	+87.3	22.7	+22.7	21.3	−15.3

Table 1: *Llama-3.2-3B-Instruct* accuracy (Acc. %) and change (Δ %) of the base model vs. after fine-tuning ($n=300$ per cell).

6 Conclusion

From Leibniz’s (1685) call to “let us calculate” to Frege’s *Begriffsschrift* (1879), the project of logic has rested on the conviction that reasoning becomes reliable only when it is translated into a formal system, where validity is established through explicit rules. The evaluations on LOGICSKILLS presented in this work suggest that conventional LLMs can achieve high accuracy on logical tasks without the explicit symbolic and model-based procedures that underpin standard logical methods. At the

³For example, rather than selecting among options 1-6, jointly fine-tuned responses often either reprint the candidate conclusions verbatim with little commentary, or devolve into repetitive step-numbered chains (since p, q (1); since q, r (2); since q, s (3) ...) that recycle the same templates without ever selecting one of the six candidate conclusions; in both cases our extractor returns *Indeterminate*.

same time, recent reasoning-tuned models exhibit a broader and more systematic logical skill profile, pointing to a promising direction for future work.

Limitations

While our study introduces a novel benchmark for formal reasoning that disentangles core logical skills and exposes failure modes that composite tasks may hide, there are several limitations that future work can address. Although the evaluation benchmark reported here is fixed in size, the underlying generation pipeline is fully programmatic and supports large-scale expansion; for example, we generate an additional 200k training instances for the PEFT experiments (§5.3) using the same infrastructure.

Logical and linguistic scope LOGICSKILLS evaluates reasoning within a deliberately restricted logical and linguistic scope. All tasks are drawn from the two-variable fragment of first-order logic without identity, excluding richer logical resources such as full first-order logic, identity, or modal operators. Consequently, results may not generalize to reasoning in more expressive logical systems. The benchmark also relies on a controlled fragment of English and a synthetically generated Carrollian language. While this design isolates logical structure, it limits linguistic diversity and does not capture many sources of difficulty present in naturally occurring language.

Evaluation procedure Model performance is additionally sensitive to prompting and evaluation protocols. In the symbolization and countermodel tasks, responses must contain logical content that can be interpreted as formulas or structures for semantic verification. However, evaluation does not depend on a fixed output template: responses are normalized, parsed, and repaired prior to verification, and cosmetic or notational variants are tolerated. Correctness is ultimately determined by logical equivalence checks and model-theoretic verification. Nevertheless, performance may still partly reflect a model’s ability to produce interpretable logical content rather than reasoning ability alone. This also applies to the PEFT results: the fine-tuning data contained only symbolization and countermodel examples, with no validity or general instruction data mixed in, so drift toward the training tasks’ response format is expected.

Behavior vs. Mechanism LOGICSKILLS evaluates reasoning behaviorally. While it reveals systematic differences across logical subskills, it does not directly address the internal representations or mechanisms by which models arrive at their answers (Chalmers, 2025; Stolfo et al., 2023). Whether reasoning-oriented models that pass all three tasks are genuinely approximating model-theoretic reasoning or achieving this through more powerful but equally opaque means remains an open question for mechanistic investigation.

Within the scope of this paper, these limitations are largely intentional design choices, reflecting a tradeoff between expressivity and reliable, solver-verifiable evaluation.

Acknowledgments

We would like to thank the anonymous reviewers for their comments and suggestions. Furthermore, we acknowledge the support for BP through the ERC Consolidator Grant DIALECT 101043235.

References

- Lewis Carroll. 1871. *Through the Looking-Glass, and What Alice Found There*. Macmillan and Co., London. Chapter 1, pp. 21–24.
- David J. Chalmers. 2025. [Propositional interpretability in artificial intelligence](#). *Preprint*, arXiv:2501.15740.
- Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2026. [Towards reasoning era: a survey of long chain-of-thought for reasoning large language models](#). *Science China Information Sciences*, 69(6):161101.
- Martin Davis and Hilary Putnam. 1960. [A computing procedure for quantification theory](#). *J. ACM*, 7(3):201–215.
- Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Gottlob Frege. 1879. *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Verlag von Louis Nebert, Halle. English translation in Jean van Heijenoort (ed.), *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*, Harvard University Press, 1967, pp. 1–82.
- Panagiotis Giadikiaroglou, Maria Lymperaiou, Giorgos Filandrianos, and Giorgos Stamou. 2024. [Puzzle solving using reasoning of large language models: A survey](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11574–11591, Miami, Florida, USA. Association for Computational Linguistics.
- Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhen-ting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. [FOLIO: Natural language reasoning with first-order logic](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.
- 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, Toronto, Canada. Association for Computational Linguistics.
- Donald Kalish and Richard Montague. 1964. *Logic: Techniques of Formal Reasoning*. Harcourt, Brace & World, New York.
- Gottfried Wilhelm Leibniz. 1685. The art of discovery. In Philip P. Wiener, editor, *Leibniz: Selections*. Charles Scribner’s Sons, New York. Translation by P. P. Wiener, 1951, p. 51; Latin in *Opusculum et Fragmenta Inédits de Leibniz*, ed. L. Couturat, Paris: Félix Alcan, 1903, p. 176.
- Yinghao Li, Haorui Wang, and Chao Zhang. 2024. [Assessing logical puzzle solving in large language models: Insights from a minesweeper case study](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, page 59–81. Association for Computational Linguistics.
- Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. 2025. [Zeblogic: On the scaling limits of LLMs for logical reasoning](#). In *Forty-second International Conference on Machine Learning*.
- Philipp Mondorf and Barbara Plank. 2024a. [Beyond accuracy: Evaluating the reasoning behavior of large language models - a survey](#). In *First Conference on Language Modeling*.
- Philipp Mondorf and Barbara Plank. 2024b. [Liar, liar, logical mire: A benchmark for suppositional reasoning in large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7114–7137, Miami, Florida, USA. Association for Computational Linguistics.
- Michael Mortimer. 1975. [On languages with two variables](#). *Mathematical Logic Quarterly*, 21(1):135–140.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. 2024. [LogicBench: Towards systematic evaluation of logical reasoning ability of large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13679–13707, Bangkok, Thailand. Association for Computational Linguistics.

Wolfgang Schwarz. 2025. Tree proof generator (tpg). <https://github.com/wo/tpg>. See also <https://www.umsu.de/trees/>.

Dana Scott. 1962. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27:477.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. [ProofWriter: Generating implications, proofs, and abductive statements over natural language](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Yuxuan Wan, Wenxuan Wang, Yiliu Yang, Youliang Yuan, Jen-tse Huang, Pinjia He, Wenxiang Jiao, and Michael Lyu. 2024. [Logicasker: Evaluating and improving the logical reasoning ability of large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2124–2155, Miami, Florida, USA. Association for Computational Linguistics.

Zonglin Yang, Xinya Du, Rui Mao, Jinjie Ni, and Erik Cambria. 2023. Logical reasoning over natural language as knowledge representation: A survey. *arXiv preprint arXiv:2303.12023*.

A Lexicon and Sentence Types

LOGICSKILLS sentences span a range of first-order constructions. Table 3 gives representative examples, covering atomic predicates, quantification (including nested and anaphoric forms), Boolean connectives, and conditionals. Each schema admits systematic surface variants (e.g., VP ellipsis, contrastive conjunctions, unless-disjunctions) that preserve the same logical structure.

Each sentence is realized in two surface languages: controlled English and a Carroll-style non-sense language, each with its own fixed lexicon. The fixed lexicons contain 3 constants (names) and 13 predicates, which bounds the maximum vocabulary size of any sentence.

B Dataset and Task Construction Pipeline

Sentence generation. All tasks are grounded in a large shared sentence inventory (100k+ sentences) generated compositionally. Each sentence is paired with a surface realization (English or Carrollian), an FO² logical form, its abstract syntax tree (AST), and a schema of abbreviation (SOA) mapping logical symbols to lexical items. English and Carroll realizations are linked by a counterpart relation preserving logical form. Sentences are labeled by syntactic type (atomic, negated, conditional, quantified, etc.) and stored in a database with fields for surface form, SOA, FO² formula, AST, language tag, and a counterpart identifier linking English and Carroll realizations.

Argument generation. Arguments are generated by sampling small premise sets (3–5 sentences) from the sentence table and applying solver-based filtering. Premises must be jointly satisfiable, and candidate conclusions are tested for entailment using Z3. To enforce non-triviality, we exclude theorems, conclusions derivable from domain constraints alone, and conclusions derivable from only a proper subset (at most 20%) of the premises. For each valid argument, five invalid arguments are constructed using alternative conclusions that fail to follow; distractors are constrained to share subject matter with the valid conclusion (same predicates and constants), preferentially match its syntactic type, and be solver-verified as non-entailed. Arguments are stored in a database with fields for premise identifiers, conclusion identifier, validity label, and language, and form the large shared pool from which validity and countermodel tasks are derived.

Task instantiation. Three task families are derived from the sentence and argument tables. Formal symbolization tasks are constructed from individual sentences paired with their SOA, with the associated FO² formula as the target. Although the fixed lexicons permit a larger vocabulary in principle, in practice sentences used to instantiate tasks involve only 3–4 predicates and 1–2 constants (4–6 symbols total). Argument validity tasks are constructed from solver-certified valid arguments, each paired with five invalid alternatives sharing the same premises. Countermodel construction tasks are constructed from solver-certified invalid arguments and require a finite model that satisfies all premises while falsifying the conclusion.

Countermodel domains. The countermodel task fixes the domain to $\{0, 1, 2\}$. Solver-certified non-entailment guarantees only that some countermodel exists, of unbounded size, so we verified that each item is solvable within this domain: we ran Schwarz’s tree proof generator (tpg; Schwarz, 2025) on every invalid argument admitted to the task and recorded the size of the countermodel returned. The maximum domain size observed across all 300 items was 3, confirming that every item is solvable within the fixed domain.

C Evaluation Procedure

All task outputs are verified automatically using Z3. The three tasks differ in how correctness is defined. In the validity task, a model is given a premise set and six candidate conclusions and must return the indices of those conclusions that follow from the premises. The gold answer is the set of such indices, which may be empty, a singleton, or multi-element; scoring is set equality, so omitting an entailed conclusion and including a non-entailed one are both errors. The symbolization and countermodel tasks admit multiple correct answers, so correctness is defined semantically rather than syntactically.

In the formal symbolization task, accuracy is defined as producing a formula logically equivalent to the target representation. Two formulas are considered equivalent if they have the same truth conditions in all structures. Evaluation therefore relies on logical equivalence checking rather than string identity. Natural-language sentences may admit multiple logically equivalent formalizations. For example, the formulas $\forall x(Dx \rightarrow \neg Bx)$ and $\neg \exists x(Dx \wedge Bx)$ are logically equivalent and therefore treated as equally correct representations of the same sentence.

Invalid arguments may admit many countermodels. The countermodel task therefore defines correctness existentially: a candidate structure is correct if it makes all premises true and the conclusion false. The task fixes a small finite domain and requires a fully specified structure over that domain. Candidate structures are normalized into domain assignments, constant interpretations, and predicate extensions prior to verification. Any structure satisfying these semantic conditions is accepted as correct.

Code for dataset generation, task construction, and evaluation is available at <https://github.com/brianrabern/LogicSkills>.

[com/brianrabern/LogicSkills](https://github.com/brianrabern/LogicSkills).

D Error Analysis Methodology

Our evaluation pipeline produced structured outputs beyond binary correctness labels, enabling systematic error analysis across all tasks. For each instance, we retained the raw model response, the extracted or parsed answer, the gold-standard answer, and task-specific diagnostic metadata from the evaluator.

Formal symbolization For *symbolization*, each response included an assessment label (e.g., *Not logically equivalent*, *Failed to parse*). Incorrect outputs were filtered and categorized by matching these labels, allowing semantic errors to be distinguished from syntactic or structural failures.

Countermodel construction For *countermodel construction*, the model checker returned a list of diagnostic error messages (e.g., *Not a countermodel*, *Missing interpretation for: X*, *Binary predicate ‘P’ must be a list of pairs*, etc.). Failures were categorized by matching error-message patterns, with additional checks performed by evaluating premises and conclusions against the proposed structure when needed.

Argument validity For *validity*, both the extracted and correct answers were represented as sets of statement indices. Set comparisons were used to distinguish incorrect selections from *super-set* errors in which the correct answer was included alongside additional invalid options.

All error analyses were performed programmatically over the full evaluation set, and the resulting error-type distributions correspond to those reported in the main text.

E Meta-Evaluation of Answer Extraction

Raw model responses were post-processed by an extractor LLM to normalize outputs into task-specific formats prior to scoring. To assess the reliability of this extraction step independently of task correctness, we conducted a meta-evaluation.

We randomly sampled 100 evaluation instances, balanced across all task types (symbolization, validity, and countermodel construction) and languages where applicable, and including both correct and incorrect model responses. For each instance, the evaluator was given the original task prompt, the model’s raw response, and the extracted answer produced by the extractor.

The meta-evaluator (GPT-4o) rendered a binary judgment—faithful or unfaithful—indicating whether the extracted answer accurately reflected the content of the raw response, independent of correctness. Five cases were flagged as potentially unfaithful.

All flagged cases, together with a random sample of 10 unflagged instances, were subsequently reviewed by a human evaluator (a meta-meta-evaluation). Of the five flagged cases, one constituted a clear extraction error, in which the model revised a predicate assignment mid-response and the extractor failed to capture the update. One case involved a normalization failure, where a malformed but correctly extracted formula was not repaired. The remaining three cases involved internally inconsistent raw responses (e.g., conflicting predicate arities or self-contradictory conclusions); in these cases, the extractor’s output was judged reasonable given the ambiguity of the source response.

After human adjudication, only one instance was deemed a clear extraction failure, with one additional case judged ambiguous, corresponding to an effective extraction accuracy of 98–99%.

F Task Prompts

Figures 3 to 5 illustrate the prompts used for each task type described in Section 3.1.

G Additional Results

Table 2 reports the complete per-model results underlying §5.

H Use of AI Assistants

We used GitHub Copilot to assist with parts of the project’s source code and Claude for proofreading and copy-editing of the manuscript. All research design, analysis, and claims are the authors’ own, and all references were verified manually.

Model	Val (Car.)	Val (Eng.)	Sym (Car.)	Sym (Eng.)	Counter
meta_llama-3.1-8B-instruct	0.46	0.45	0.14	0.14	0.00
meta_llama-3.1-70B-instruct	0.84	0.86	0.66	0.69	0.12
qwen2.5-math-72B-instruct	0.80	0.80	0.80	0.80	0.15
anthropic_claude-3.7-sonnet	0.96	0.94	0.71	0.76	0.42
openai/gpt-4o	0.87	0.85	0.74	0.72	0.10
google_gemini-2.5-flash	0.96	0.94	0.76	0.79	0.13
deepseek_chat	0.81	0.76	0.80	0.81	0.35
microsoft_phi-4	0.80	0.73	0.51	0.57	0.14
qwen3-32B	0.95	0.97	0.82	0.85	0.89
openai/o3	0.99	0.99	0.97	0.94	0.99

Table 2: Accuracy across tasks and models (300 problems per setting). 95% Wilson intervals are at most ± 5.6 points wide.

ARGUMENT VALIDITY PROMPT

Your task is to solve a logical reasoning problem. Use any approach you find effective, but clearly and explicitly state your final answer.

Task

Consider the following situation:

Everything is a tove, a borogove, or a rath (exclusively), and there's at least one of each. Zindle or Bungo will whiffle. Only toves will whiffle. Every rath chortled at Bungo. If Zindle will whiffle, then every rath is mimsy only if every borogove chortled at Bungo.

Which, if any, of the following statements must be true in this situation?

1. Zindle and Bungo will whiffle.
2. Every tove will whiffle.
3. Not all toves will whiffle.
4. A tove will whiffle.
5. No toves will whiffle.
6. Zindle will whiffle.

Figure 3: Illustrative instance of the *Argument Validity* task.

FORMAL SYMBOLIZATION PROMPT

Your task is to translate the provided sentence into formal predicate logic, using the abbreviations provided.

Instructions

- Use only the abbreviations given.
- Return a single well-formed formula in standard predicate logic syntax.
- Use standard logical symbols:
 - Quantifiers: $\forall$, $\exists$
 - Connectives: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
 - Do not include any explanation or extra text—just the formula.

Example

Sentence: Every linguist admires Charlie.

Abbreviations:

- L: “[1] is a linguist”
- R: “[1] admires [2]”
- c: “Charlie”

Translation: $\forall x(Lx \rightarrow Rxc)$

Task

Sentence: Not all raths will whiffle and no toves chortled at Bungo.

Abbreviations:

- O: “[1] is a rath”
- F: “[1] will whiffle”
- M: “[1] is a tove”
- P: “[1] chortled at [2]”
- c: “Bungo”

Figure 4: Illustrative instance of the *Formal Symbolization* task.

COUNTERMODEL CONSTRUCTION PROMPT

Show that the provided argument is invalid by giving a countermodel—one where all premises are true and the conclusion is false.

Instructions

1. Provide assignments for all constants and predicates used in the argument.
2. Respect predicate arity:
 - Monadic predicates take one argument (e.g., Mx).
 - Binary predicates take two arguments (e.g., Pxy).
3. Use the fixed domain $[0, 1, 2]$.

Required Format

- Domain: $[0, 1, 2]$
- Constants: “a”: 0
- Monadic predicates: “F”: $[0, 2]$
- Binary predicates: “R”: $[[0, 1], [2, 0]]$

Argument

$$\begin{aligned} & (\forall x (Mx \vee (Nx \vee Ox)) \wedge (\neg \exists x (Mx \wedge Nx) \wedge (\neg \exists x (Mx \wedge Ox) \wedge \neg \exists x (Nx \wedge Ox)))), \\ & (\exists x Mx \wedge (\exists x Nx \wedge \exists x Ox)), \\ & \neg \exists x \exists y ((Mx \wedge My) \wedge Pxy), \\ & (\neg \exists x (Ox \wedge Qxc) \rightarrow Mb), \\ & (\neg \exists x (Nx \wedge Rxc) \rightarrow \exists x (Nx \wedge \neg Rxb)), \\ & (\forall x (Ox \rightarrow Rxc) \rightarrow (Mb \rightarrow \forall x (Nx \rightarrow Pxc))), \\ & (Pcc \wedge \forall x (Ox \rightarrow Rxa)) \\ & \models Mc \end{aligned}$$

Figure 5: Illustrative instance of the *Countermodel Construction* task.

Sentence type	English example	Carroll example
atomic_monadic	Hazel drank.	Bungo gyred.
atomic_dyadic	Hazel kicked Lewis.	Bungo galumphed over Rafin.
negation	Lewis isn't happy.	Rafin isn't uffish.
quantified_universal_affirmative	Every monkey is asleep.	Every rath is mimsy.
quantified_particular_affirmative	A donkey will run.	A tove will whiffle.
quantified_universal_negative	No humans chased Alfred.	No borogoves snicker-snacked Zindle.
quantified_particular_negative	Not all donkeys chased Alfred.	Not all toves snicker-snacked Zindle.
quantified_only_restrictor	Only humans will run.	Only borogoves will whiffle.
quantified_name_all	Hazel chased every human.	Bungo snicker-snacked every borogove.
quantified_name_some	Alfred chased a donkey.	Zindle snicker-snacked a tove.
quantified_all_all	Every monkey kicked every donkey.	Every rath galumphed over every tove.
quantified_all_all_all	Every donkey saw every donkey that saw every monkey.	Every tove chortled at every tove that chortled at every rath.
quantified_all_all_back	Every donkey chased every monkey that chased it.	Every tove snicker-snacked every rath that snicker-snacked it.
quantified_all_some	Every monkey kicked a human.	Every rath galumphed over a borogove.
quantified_all_some_back	Every human saw a donkey that saw it.	Every borogove chortled at a tove that chortled at it.
quantified_some_all	A monkey chased every human.	A rath snicker-snacked every borogove.
quantified_some_all_back	A monkey saw every human that saw it.	A rath chortled at every borogove that chortled at it.
quantified_some_some	A human chased a donkey.	A borogove snicker-snacked a tove.
quantified_some_some_back	A donkey kicked a monkey that kicked it.	A tove galumphed over a rath that galumphed over it.
quantified_some_some_some	A human kicked a human that kicked a monkey.	A borogove galumphed over a borogove that galumphed over a rath.
quantified_no_all	No humans kicked every human.	No borogoves galumphed over every borogove.
quantified_no_some	No humans chased a human.	No borogoves snicker-snacked a borogove.
quantified_no_some_back	No donkeys kicked a monkey that kicked it.	No toves galumphed over a rath that galumphed over it.
quantified_rev_some_all	There is a donkey that every monkey kicked.	There is a tove that every rath galumphed over.
quantified_rev_no_all	There is not a monkey that every human chased.	There is not a rath that every borogove snicker-snacked.
quantified_some_self	A donkey kicked itself.	A tove galumphed over itself.
conjunction_simple	Hazel saw Lewis and not all monkeys kicked Lewis.	Bungo chortled at Rafin and not all raths galumphed over Rafin.
disjunction_simple	Alfred drank or a monkey kicked Alfred.	Zindle gyred or a rath galumphed over Zindle.
conditional_if_then	If Hazel is a human, then Alfred isn't a human.	If Bungo is a borogove, then Zindle isn't a borogove.
biconditional_just_in_case	Hazel is happy just in case every donkey chased Lewis.	Bungo is uffish just in case every tove snicker-snacked Rafin.

Table 3: Sentence types used in LOGICSKILLS with representative English and Carroll examples.