Beyond Accuracy: Introducing a Symbolic-Mechanistic Approach to Interpretable Evaluation

Reza Habibi*, **Darian Lee***,
Magy Seif El-Nasr
University of California, Santa Cruz
606 Engineering Loop, Santa Cruz, CA 95064 USA
rehabibi@ucsc.edu, daeilee@ucsc.edu

Abstract

Accuracy-based evaluation cannot reliably distinguish genuine generalization from shortcuts like memorization, leakage, or brittle heuristics, especially in small-data regimes. In this **position paper**, we argue for mechanism-aware evaluation that combines task-relevant symbolic rules with mechanistic interpretability, yielding algorithmic pass/fail scores that show exactly where models generalize versus exploit patterns. We demonstrate this on NL-to-SQL by training two identical architectures under different conditions: one without schema information (forcing memorization), one with schema (enabling grounding). Standard evaluation shows the memorization model achieves 94% field-name accuracy on unseen data, falsely suggesting competence. Our symbolic-mechanistic evaluation reveals this model violates core schema generalization rules, a failure invisible to accuracy metrics.

Introduction and Background

Standard NLP evaluation relies on held-out test sets with metrics like exact match, F1, or BLEU (Papineni et al. 2002; Nakache, Metais, and Timsit 2005; Gehrmann, Clark, and Sellam 2023). However, recent work reveals these methods cannot reliably distinguish memorization from generalization. Contamination is widespread, with benchmarks like QNLI showing over 50% contamination in popular pretraining corpuses (Lee et al. 2022; Dodge et al. 2021), and models exploit spurious heuristics even on clean data (Gururangan et al. 2018; McCoy, Pavlick, and Linzen 2019; Kim et al. 2025). Alternative methods like LLM-as-a-judge suffer from self-preference bias and adversarial vulnerabilities (Chen et al. 2024; Raina, Liusie, and Gales 2024). We argue these failures stem from a fundamental problem: current evaluation measures surface-level performance rather than underlying computational mechanisms. We propose mechanistic interpretability (MI) evaluation that directly examines whether models use correct algorithms.

Contamination and Exploitable Heuristics

Contamination research has demonstrated the scope of this issue. (Zhang et al. 2024) found 0.32 correlation between memorization probability and performance on GSM8K variants, while (Li and Flanigan 2024) found 0.88 correlation between exact-match generation without context and test performance. (Aiyappa et al. 2023) observed ChatGPT's benchmark performance improved after papers evaluating it were published, indicating RLHF-driven contamination. Critically, (Samuel, Zhou, and Zou 2025) found alarmingly low agreement between contamination detection methods, indicating no reliable standard exists, implying that mitigation and detection of contamination before testing is not a viable option. Additionally, modern training pipelines with RLHF on user data and continuous web scraping create ongoing contamination pathways that cannot be eliminated (Zhang et al. 2024; Aiyappa et al. 2023).

Even with pristine data, models exploit spurious patterns. (Gururangan et al. 2018) achieved 67% SNLI accuracy using only hypothesis sentences by exploiting annotation artifacts. (Kim et al. 2025) showed mechanistically that benchmarks often fail to test their advertised skills, instead rewarding alternative mechanisms. For tasks with small datasets, common in low-resource languages and specialized domains, test sets lack statistical power to separate genuine competence from pattern matching.

From Grokking to Verification

Research on grokking demonstrates that generalization corresponds to distinct internal structure (Power et al. 2022; Nanda et al. 2023; Liu et al. 2022). (Nanda et al. 2023) showed that generalization circuits form during memorization before full utilization, suggesting mechanistic probing can detect whether models operate in memorization- versus generalization-dominant regimes. However, grokking insights remain unapplied to real NLP evaluation.

This motivates symbolic verification. Symbolic AI provides frameworks for expressing correctness properties as semantically meaningful rules (Newell, Shaw, and Simon 1957, 1959; McCarthy 1995; Xie, Kersting, and Neider 2022). By checking whether internal computations match expected generalization patterns, symbolic rules enable more rigorous evaluation than accuracy alone.

Proposal

Our approach employs "non-negotiable rules" in symbolic logic describing properties any circuit reliably solving the

*These authors contributed equally.

task must satisfy. Given task T, the user specifies rules R = {r_1, ..., r_k} capturing essential requirements at the level of task semantics: what information must be used, what invariances must hold. Rules must be verifiable via common mechanistic interventions like activation patching, logit lens analysis, or attention visualization.

For each example in a small evaluation set, we test whether rules are satisfied, assigning pass/fail scores. The aggregate percentage indicates how consistently the model uses proper generalization circuits.

This paper is a *position paper* advocating mechanism-aware evaluation for tasks with a well-defined intended algorithm (e.g., grounding, retrieval, or parsing). We do not claim applicability to open-ended or creative generation tasks, where multiple qualitatively different algorithms may be equally valid and no single mechanistic criterion is appropriate.

Case Study

We compare our evaluation framework to standard accuracy on a simple NL-to-SQL task, focusing on schema grounding as the core algorithm. We train two models with identical architectures but different training conditions: one without schema information (NO_Schema), forcing reliance on shallow heuristics, and one with schema (Schema), enabling genuine generalization. We show that standard evaluation suggests roughly equal capabilities, whereas symbolic mechanistic evaluation reveals the NO_Schema model's lack of true generalizability.

Dataset Our case study employs the TinySQL CS1 Synonyms dataset (Harrasse et al. 2025). Each example consists of an English prompt (`english_prompt`), a database schema provided as a `CREATE TABLE` statement (`create_statement`), and the target SQL query (`sql_statement`). Critically, the schema uses a synonym of the column name mentioned in the English prompt,[1] ensuring that correct SQL cannot be generated from natural language alone without schema grounding.

The dataset contains only simple queries following the pattern `SELECT x FROM y`. Given this structural simplicity, we focus evaluation on schema grounding, the ability to correctly map English column names to their schema synonyms, as the core algorithmic capability. The remaining query structure follows a predictable template across all examples.

Training Conditions: We finetune two models based on TinyStories-33M (Eldan and Li 2023), with identical architecture and optimization settings, following (Harrasse et al. 2025). The models differ only in schema presence during training. The baseline was pretrained solely on TinyStories-33M short stories with no NL-to-SQL examples, ensuring no task contamination (Eldan and Li 2023).

The **schema** condition includes the `create_statement` in context for every training example. The **NO_Schema** condition omits it entirely while keeping the same target SQL, forcing reliance on shallow heuristics rather than schema consultation. The column and table names in the English prompt are synonyms rather than exact matches, thus models benefit from using schema information when available (Harrasse et al. 2025).

We use a single standardized prompt format: `### Instruction: {english_prompt} ### Context: {create_statement or empty} ###Response: {sql_statement}`. Decoding begins after "### Response:", and predicted SQL is compared against `sql_statement`.

Standard Accuracy-based Testing: We compare our approach to standard evaluation on the TinySQL CS1 Synonyms test split, reporting exact match accuracy (predictions identical to gold SQL) and field name accuracy (correct table/column names as a ratio of total gold field names, excluding SQL keywords). Each model is evaluated under two configurations: schema (schema included in test prompt) and NO_Schema (schema excluded).

Symbolic Mechanistic Evaluation

We evaluate 100 matched prompt pairs where each has a *clean* prompt with the correct schema field and a *corrupted* version with an alternative word, holding format constant.

We test five corruption types to probe schema grounding. **DB-synonyms** replace column names with semantically related alternatives that also appeared as column names in training ("price"/"cost"), testing whether the model maps semantic equivalents within the trained schema. **Non-DB synonyms** use semantically related words that were never column names ("mouse"/"rat"), testing whether this semantic flexibility generalizes. The remaining three test whether the model is sensitive to schema violations at all, measuring whether it can detect and react when an unexpected word appears. **DB-scramble** pairs unrelated words that both appeared as column names ("price"/"brand"), **Non-DB scramble** pairs unrelated words that were not column names ("mouse"/"car"), and **Super-scramble** mixes column names with non-column words ("price"/"rat"). If the model detects these scrambled corruptions, it demonstrates active tracking of the schema.

Symbolic Rules To evaluate whether a schema-checking circuit exists, we adopt a symbolic rule-based approach that decomposes this question into three testable conditions. First, we test causal sensitivity: does corrupting the schema token change which answer the model prefers? Second, we test localization: can we recover this effect by patching activations at the schema token position? Third, we test consistency: do multiple examples recover through the same layer, suggesting a reusable mechanism? We call these rules R1 through R3. Because R3 must only pass if R1 and R2 pass, we denote that a pass on R3 is analogous to all conditions being met. We measure preference for the correct token using logit-difference $\text{logit\_diff}(x; i) = \ell(x)_{\text{correct}_i} - \ell(x)_{\text{incorrect}_i}$, where $\ell(x)$ denotes model logits at the final position. Let x_i^{clean} and x_i^{corr} denote clean and corrupted prompts differing

[1] E.g., schema `CREATE TABLE backup (website TEXT)`, prompt "Pull up url from safekeeping copy", target `SELECT website FROM backup`.

only in the schema token. We define the schema-sensitivity gap as

$$\Delta_i = \text{logit\_diff}(x_i^{\text{clean}}; i) - \text{logit\_diff}(x_i^{\text{corr}}; i).$$

A positive Δ_i indicates the model causally depends on the schema token when selecting answers.

Activation Patching Protocol To test whether Δ_i can be recovered from specific layers, we perform residual-stream activation patching at the schema token position (p). We cache activations from x_i^{clean}, then for each layer $L \in \{0, \ldots, L_{\max}\}$, re-run x_i^{corr} while replacing the residual activation at (L, p) with the cached clean activation, yielding $\text{shift}_i(L) = \text{patched\_diff}_i(L) - \text{corr\_diff}_i$. We define the best signed recovery and the layer achieving it as

$$\text{best\_recovery}_i = \max_L \max\big(0,\ \text{sign}(\Delta_i) \cdot \text{shift}_i(L)\big)$$

$$L_i^* = \text{argmax}_L \max\big(0,\ \text{sign}(\Delta_i) \cdot \text{shift}_i(L)\big)$$

We cap recovery at $|\Delta_i|$ and define fractional recovery as $\text{RecFrac}_i = \min(\text{best\_recovery}_i, |\Delta_i|)/|\Delta_i|$. We denote TopLayers_i as the set of layers achieving $\geq$90% recovery, recognizing potential superposition and redundant circuits (Elhage et al. 2022).

Hierarchical Rule System Intuitively, our evaluation asks three questions: (1) does changing the schema token causally affect the model's answer preference, (2) can this effect be localized to a specific internal computation via patching, and (3) is the same computation reused across inputs? The following definitions formalize these questions using logit-difference measurements and residual-stream interventions. Starting with a top-level claim about circuit existence, we decompose into per-example causal tests (R1, R2, R3).

Let $\tau_{\text{gap}} \geq 0$ be a minimum gap threshold, and $\alpha \in (0, 1]$ be the required recovered fraction.

R1 (schema_sensitivity). If the schema token is corrupted, the model's preference changes by at least τ_{gap}:

$$R1_i \ = \ \mathbb{I}\big[|\Delta_i| \geq \tau_{\text{gap}}\big].$$

This tests causal dependence on the schema token. Examples with $|\Delta_i| < \tau_{\text{gap}}$ indicate the model ignores schema information. Passing R1 shows the model's answer preference depends on which schema token is present.

R2 (recovery_efficacy). If R1 holds and the clean schema token activation is patched at position (p), then at least α fraction of the gap is recovered:

$$R2_i \ = \ R1_i \ \wedge \ \mathbb{I}\big[\text{RecFrac}_i \geq \alpha\big].$$

This tests whether the causal effect localizes to a specific layer. Measuring recovery as a fraction (RecFrac_i) assesses mechanism presence independently of gap magnitude. Passing R2 shows a single layer at the schema position is sufficient to recover preference, indicating a localized circuit. Due to superposition or redundant circuits, multiple layers can achieve $\geq$90% recovery, captured in TopLayers_i.

R3 (circuit_reusability). If R1 and R2 hold and the most common high-recovery layer across all examples is L^*, then $L^* \in \text{TopLayers}_i$:

$$R3_i \ = \ R1_i \ \wedge \ R2_i \ \wedge \ \mathbb{I}\big[L^* \in \text{TopLayers}_i\big].$$

This tests whether recovery uses a consistent layer across examples. Passing R3 shows the model uses the same computational pathway for different queries, indicating reuse of a consistent computational pathway, as opposed to relying on input-specific heuristics.

We run the identical evaluation pipeline on our schema-trained model and NO_Schema baseline model, and compare per-example pass rates ($\frac{1}{N}\sum_i R3_i$) under the same prompt set and thresholds. Remember that R3 can only pass if R1 and R2 pass as well.

This framework yields an example-level pass rate estimating proximity to complete generalization. Decoupling gap significance (R1) from recovery quality (R2) measures mechanism presence independently of task difficulty or corruption strength. The consistency metric (R3) distinguishes position-specific effects from genuine circuit reuse, providing stronger evidence for localized computation.

Results

Standard Accuracy

The TinySQL CS1 synonyms test-set results reveal a critical limitation of surface-level evaluation, as the model trained without schema achieves 93.5% table/column accuracy even when schema is withheld at test time. Since the model has no access to schema information, this high performance can only result from exploiting spurious patterns in the data. Although the schema-trained model performs better (99.1% with schema present), these metrics alone cannot distinguish whether the model genuinely uses schema information or simply memorizes training patterns (Table 1).

Train Schema	Eval Schema	Exact Match	Field Acc
✗	✗	**72.1%**	**93.5%**
✗	✓	10.4%	83.6%
✓	✗	0.0%	40.4%
✓	✓	**94.0%**	**99.1%**

Table 1: Standard accuracy across model configurations. The NO_Schema model achieves 93.5% field-name accuracy by exploiting patterns, demonstrating traditional evaluation's unreliability.

Symbolic Mechanistic Results

On our 100 controlled prompt pairs, the schema-trained model exhibited significantly larger sensitivity to schema corruption (mean $\Delta_{\text{schema}} = 1.88$ [95% CI: 1.44, 2.32]) compared to the NO_Schema baseline (mean $\Delta_{\text{NO\_Schema}} = 0.65$ [95% CI: 0.39, 0.90]), yielding a training effect of $+1.23$ [95% CI: 0.79, 1.67] logits (190% increase). Per-category analysis (Table 2) shows the effect is largest for

scrambled corruptions (Super-Scramble: $+2.35$, Non-DB-Scramble: $+1.61$) and smaller for synonyms ($+0.53$ to $+0.57$), suggesting the model partially accepts semantically related substitutions.

Applying our symbolic rule framework (R1: $\Delta > 0.4$; R2: recovery $\geq 90\%$ of $|\Delta|$; and R3: $L^* \in \text{TopLayers}_i$) **the schema model achieved 76% overall PASS rate (76/100 examples) versus 59% for the NO_Schema model**[2](Table 2). Category-specific PASS rates ranged from 55% (DB-Synonyms) to 95% (Non-DB-Scramble) for the schema model, consistently exceeding the NO_Schema baseline across all categories. The consistent advantage across all five categories and large effect sizes with non-overlapping CIs support the framework's discriminative power.

Passing examples exhibited strong circuit consistency: among 76 schema-model examples satisfying R3 (indicating all three rules passed), Layers 0–2 collectively accounting for 89%. In contrast, the NO_Schema model showed distributed processing (modal layer usage: 34%). This layer-level convergence in combination with high patching recovery rates provides evidence for a localized, reusable schema-checking circuit in the schema-trained model.

Category	**Effect** ($\Delta_S - \Delta_N$)	**PASS**$_S$	**PASS**$_N$
DB-Synonyms	+0.53	55%	25%
DB-Scramble	+0.51	75%	70%
Non-DB-Syn	+0.57	80%	45%
Non-DB-Scr	+2.28	95%	85%
Super-Scr	+0.77	75%	70%
Overall	**+0.93**	**76%**	**59%**

Table 2: Per-category results across 100 examples (20 per category).

Discussion

Our results show that mechanistic evaluation distinguishes algorithmic learning from pattern-matching more effectively than standard metrics. The schema model achieved 76% pass rate while the NO_Schema baseline reached 59%, a 17-percentage-point gap (compared to a 5-percentage-point gap in standard accuracy testing). This inversion shows that surface metrics dramatically underestimate the difference between genuine algorithmic learning and pattern exploitation.

The symbolic mechanistic evaluation distinguishes these cases where standard metrics fail. The symbolic rule system provides interpretable diagnostics where R1 failures indicate the model ignores critical input features, R2 failures reveal distributed rather than localized computation, and R3 failures show inconsistent processing across examples.

Current benchmarks face widespread data leakage with no reliable detection consensus. Models exploit spurious patterns even in clean test sets. Our NO_Schema baseline demonstrates this by achieving 93.5% field-name accuracy despite lacking the intended algorithm. Standard evaluation cannot detect this failure mode, however our symbolic-mechanistic approach correctly identifies this limitation.

[2]Alternative thresholds (R1: $\Delta \in \{0.25, 0.3\}$; R2/R3: $\geq 80\%, 95\%$) showed schema model outperforming baseline by 10-20pp across all combinations, suggesting rules are robust to threshold noise.

Conclusion and Future Work

This work introduces a mechanism-aware evaluation framework combining symbolic rules with mechanistic interpretability to separate genuine generalization from pattern exploitation. Applied to NL-to-SQL, we show that a model reaching 93.5% field-name accuracy achieves only 59% on mechanism consistency checks, demonstrating that standard metrics cannot reliably assess whether models use intended algorithms. The hierarchical rule system provides interpretable diagnostics and serves as a template for tasks with well-defined algorithmic primitives such as retrieval, grounding, or structured parsing.

Three limitations suggest productive future directions. First, our rules depend on hyperparameters that need careful tuning. Second, activation patching risks distribution shifts that may enable recovery through confounding pathways; while we use convergent validation across multiple intervention types, future work should triangulate with path patching, causal tracing, or ablation studies. Third, symbolic rules become hard to define for complex tasks lacking clear algorithmic primitives. Our framework suits targeted capability assessment more than holistic evaluation. More broadly, mechanism-aware evaluation surfaces failure modes invisible to accuracy metrics. As model contamination grows widespread and pattern matching grows more sophisticated, verifying that models implement intended algorithms becomes essential. Future benchmarks should require mechanism certification alongside accuracy reporting, particularly in high-stakes domains where reasoning transparency matters.

References

Aiyappa, R.; An, J.; Kwak, H.; and Ahn, Y.-y. 2023. Can we trust the evaluation on ChatGPT? In Ovalle, A.; Chang, K.-W.; Mehrabi, N.; Pruksachatkun, Y.; Galystan, A.; Dhamala, J.; Verma, A.; Cao, T.; Kumar, A.; and Gupta, R., eds., *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, 47–54. Toronto, Canada: Association for Computational Linguistics.

Chen, G. H.; Chen, S.; Liu, Z.; Jiang, F.; and Wang, B. 2024. Humans or LLMs as the Judge? A Study on Judgement Bias. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 8301–8327. Miami, Florida, USA: Association for Computational Linguistics.

Dodge, J.; Sap, M.; Marasović, A.; Agnew, W.; Ilharco, G.; Groeneveld, D.; Mitchell, M.; and Gardner, M. 2021. Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 1286–1305. Online and Punta Cana, Do-

minican Republic: Association for Computational Linguistics.

Eldan, R.; and Li, Y. 2023. TinyStories: How Small Can Language Models Be and Still Speak Coherent English? arXiv:2305.07759.

Elhage, N.; Hume, T.; Olsson, C.; Schiefer, N.; Henighan, T.; Kravec, S.; Hatfield-Dodds, Z.; Lasenby, R.; Drain, D.; Chen, C.; Grosse, R.; McCandlish, S.; Kaplan, J.; Amodei, D.; Wattenberg, M.; and Olah, C. 2022. Toy Models of Superposition. ArXiv:2209.10652 [cs.LG].

Gehrmann, S.; Clark, E.; and Sellam, T. 2023. Repairing the Cracked Foundation: A Survey of Obstacles in Evaluation Practices for Generated Text. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 14007–14027. Toronto, Canada: Association for Computational Linguistics.

Gururangan, S.; Swayamdipta, S.; Levy, O.; Schwartz, R.; Bowman, S.; and Smith, N. A. 2018. Annotation Artifacts in Natural Language Inference Data. In Walker, M.; Ji, H.; and Stent, A., eds., *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 107–112. New Orleans, Louisiana: Association for Computational Linguistics.

Harrasse, A.; Quirke, P.; Neo, C.; Nathawani, D.; Marks, L.; and Abdullah, A. 2025. TinySQL: A Progressive Text-to-SQL Dataset for Mechanistic Interpretability Research. In Christodoulopoulos, C.; Chakraborty, T.; Rose, C.; and Peng, V., eds., *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 29256–29284. Suzhou, China: Association for Computational Linguistics. ISBN 979-8-89176-332-6.

Kim, D.; Shim, G.; Chun, Y.; Kim, M.; and Seo, M. 2025. Benchmark Profiling: Mechanistic Diagnosis of LLM Benchmarks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 14226–14246. Miami, Florida, USA: Association for Computational Linguistics.

Lee, K.; Ippolito, D.; Nystrom, A.; Zhang, C.; Eck, D.; Callison-Burch, C.; and Carlini, N. 2022. Deduplicating Training Data Makes Language Models Better. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 8424–8445. Dublin, Ireland: Association for Computational Linguistics.

Li, C.; and Flanigan, J. 2024. Task contamination: Language models may not be few-shot anymore. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 18471–18480.

Liu, Z.; Kitouni, O.; Nolte, N.; Michaud, E. J.; Tegmark, M.; and Williams, M. 2022. Towards understanding grokking: an effective theory of representation learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713871088.

McCarthy, J. 1995. *Programs with common sense*, 479–492. USA: American Association for Artificial Intelligence. ISBN 0262621010.

McCoy, R. T.; Pavlick, E.; and Linzen, T. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In Korhonen, A.; Traum, D.; and Màrquez, L., eds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3428–3448. Florence, Italy: Association for Computational Linguistics.

Nakache, D.; Metais, E.; and Timsit, J. F. 2005. Evaluation and NLP. In *International Conference on Database and Expert Systems Applications*, 626–632. Springer.

Nanda, N.; Chan, L.; Lieberum, T.; Smith, J.; and Steinhardt, J. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*.

Newell, A.; Shaw, J. C.; and Simon, H. A. 1957. Empirical explorations of the logic theory machine: a case study in heuristic. In *Papers Presented at the February 26-28, 1957, Western Joint Computer Conference: Techniques for Reliability*, IRE-AIEE-ACM '57 (Western), 218–230. New York, NY, USA: Association for Computing Machinery. ISBN 9781450378611.

Newell, A.; Shaw, J. C.; and Simon, H. A. 1959. Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing*, 256–264. UNESCO.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311–318.

Power, A.; Burda, Y.; Edwards, H.; Babuschkin, I.; and Misra, V. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*.

Raina, V.; Liusie, A.; and Gales, M. 2024. Is LLM-as-a-Judge Robust? Investigating Universal Adversarial Attacks on Zero-shot LLM Assessment. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 7499–7517. Miami, Florida, USA: Association for Computational Linguistics.

Samuel, V.; Zhou, Y.; and Zou, H. P. 2025. Towards data contamination detection for modern large language models: Limitations, inconsistencies, and oracle challenges. In *Proceedings of the 31st International Conference on Computational Linguistics*, 5058–5070.

Xie, X.; Kersting, K.; and Neider, D. 2022. Neuro-Symbolic Verification of Deep Neural Networks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, 3622–3628.

Zhang, H.; Da, J.; Lee, D.; Robinson, V.; Wu, C.; Song, W.; Zhao, T.; Raja, P.; Zhuang, C.; Slack, D.; et al. 2024. A careful examination of large language model performance on grade school arithmetic. *Advances in Neural Information Processing Systems*, 37: 46819–46836.