An Explanation In Support Of Neuro-Symbolic Language Models for Scaling Algorithmic Reasoning

Terry Tong [1] **Yu Feng** [1] **Surbhi Goel** [1] **Dan Roth** [1]

Abstract

Large language models can solve algorithmic problems either through direct natural language reasoning or by generating executable code delegated to an external solver. However, little theoretical progress has been made on explaining *why* code-based approaches consistently outperform natural language reasoning.

We introduce a three-arm framework that makes this comparison tractable by introducing an intermediary step—code generation with LLM-based execution—enabling pairwise theoretical analysis via Bayesian inference and information theory.

Empirically, we demonstrate on arithmetic, dynamic programming, and integer linear programming tasks that code execution achieves 78% accuracy versus 30% for code simulation and 21% for natural language reasoning across Deepseek and Gemma models ($p < 0.05$, Friedman test). Theoretically, we prove that code representations yield higher mutual information with the target algorithm, leading to at least 6% lower Bayes error than natural language.

These results inform the design of compositional AI systems, providing principled guidance on when to use tool-augmented versus monolithic reasoning for algorithmic tasks. Our framework offers a unified perspective on the tool-use versus direct-reasoning tradeoff.

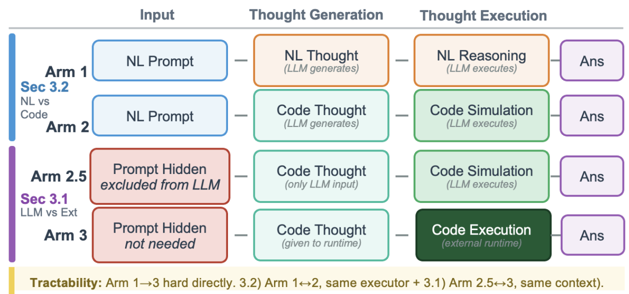

Figure 1. Bayesian Inference model showing the *three arms methodology* in **??**. Given an algorithmic problem, we may split it into two steps (1) Translate (2) Execute. The (1) translation $\in \{\text{Code}, \text{NL}\}$. Then (2) execution $\in \{\text{LLM Reasoning}, \text{Solver Execution}\}$. We have three pairs, Arm 1: $\{\text{NL Gen}, \text{LLM Reasoning}\}$, Arm 2: $\{\text{Code Gen}, \text{LLM Reasoning}\}$, Arm 3: $\{\text{Code Gen}, \text{Solver Execution}\}$. Typically, the problem is tackled by comparing Arm 1 and Arm 3 in neuro-symbolic literature, which is intractable theoretically, and uncontrolled since multiple variables are changing. By introducing Arm 2, the problem becomes tractable. In the diagram, the *shaded* circles correspond to observed R.V. and *white* correspond to unobserved. The notation for R.V.s is correspondingly used in Section 2.

1. Introduction

Consider the algorithmic task of computing arithmetic operations encoded in natural language. We wish to compute:

$$p(\texttt{Y=3}|\texttt{X=What is one plus two?})$$

The language model forward pass could decide (1) to generate the solution directly with natural language reasoning (Wei et al., 2022) (2) translate the problem into code and use a solver (Gao et al., 2023). This paper provides empirical evidence that Arm 1 < Arm 3 quantified by end-task accuracy. A body of work shows that this pipeline is generally effective (Lyu et al., 2023; Pan et al., 2023), further evidenced by the empirical success of tool-use[1]. However, little progress has been made on explaining *why* solver-based tools lead to higher end-task accuracy than natural language reasoning.

One might be tempted to prove a statistical advantage by showing that sample complexity to learn code is less than natural language because code is structured, but will quickly find that this problem becomes intractable due to the hardness of capturing natural language under a mathematical framework. The same problem arises when attempting to use approximation theory to provide evidence that DNNs

[*]Equal contribution [1]University of Pennsylvania. Correspondence to: Terry Tong <tongt1@seas.upenn.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

[1]Here we primarily refer to solver-based tool-use as opposed to knowledge-intensive tool-use like RAG

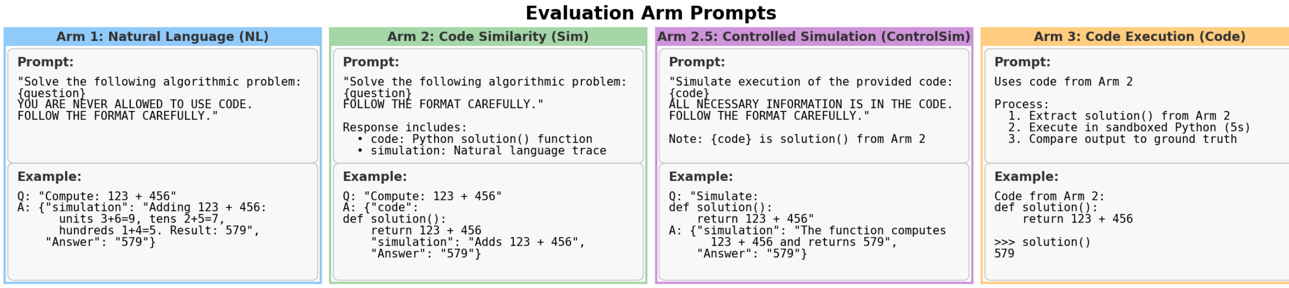

Figure 2. Prompt templates for the three-arm evaluation framework. Arm 1 instructs the model to reason purely in natural language without code. Arm 2 instructs the model to generate code and simulate its execution. Arm 3 uses the same code generation prompt but executes the output in a Python runtime rather than simulating.

can better learn compositional or structured languages (like code) than natural language. Broadly speaking, the problem is challenging because the inputs and outputs are different, one is a structured language, the other is an unstructured language. This drastically complicates comparison. Or, one might be tempted to show a computational advantage by showing that code unlocks a new level of expressivity (Merrill & Sabharwal, 2023). However, one will find that whether using a solver or not will not overcome the hardness of the problem. E.g., we can prove by contradiction that using a solver will not allow us to better solve NP-Hard problems, unless P=NP.

As a solution, we leverage a Bayesian Inference paradigm to reason about the two different settings (Xie et al., 2021). Doing so, enables us to break down the algorithmic reasoning pipeline into two distinct phases (1) Translation $\in$ $\{\mathrm{Code}, \mathrm{NL}\}$ (2) Execution $\in \{\mathrm{LLM\ Reasoning}, \mathrm{Solver\ Execution}\}$ (Lyu et al., 2023; Pan et al., 2023). Enumerating the valid combinations, we obtain three pairs, Arm 1: $\{\mathrm{NL\ Gen}, \mathrm{LLM\ Reasoning}\}$, Arm 2: $\{\mathrm{Code\ Gen}, \mathrm{LLM\ Reasoning}\}$, Arm 3: $\{\mathrm{Code\ Gen}, \mathrm{Solver\ Execution}\}$. Introducing Arm 2 makes the problem tractable.

Empirically, this framework enables controlled comparisons. A rigorous comparison has not been instantiated because it is hard to control the experiment and determine what representation is actually being used—whether code, natural language, or something else. We overcome this by verbalizing the representation using Chain-of-Thought. We provide statistical evidence for the alternative hypothesis Arm 3 $>$ Arm 2 $>$ Arm 1 on (Gemma, Deepseek, Llama), we demonstrate that generating code and executing leads to 78% accuracy on (ILP, DP, Arithmetic) tasks, over 30% for code simulation and just over 20% for natural language reasoning across 5 models. Our results are computed over 5 seeds, and a Friedman Chi-square gives results a test statistic of 9277.32 and 8369.34 for deepseek and llama, enabling us to reject the null in favor of the alternative.

Theoretically, we first compare Arm 1 $<$ Arm 2. Bayesian Inference shows us that the LLM implicitly does multi-class classification to the right algorithm. We utilize information theory to capture natural language and code under the same framework, forgoing using grammars or other mathematical frameworks that are intractable. We reduce the comparison of Bayes Error to that of comparing cross-entropy. A intermediate step using mutual information makes the proof interpretable: we prove that the mutual information between the CoT and the final answer is higher when conditioned on code representations over natural language representations. We variationally lower bound the mutual information using cross-entropy of a proposal distribution parameterized by a logistic regression. Since we only care about orderings, we subtract to overcome the intractability of estimating the differential entropy, reducing the comparison of mutual information to that of cross-entropy. The cross-entropy is measured empirically using logistic regression on TF-IDF features and Bert-base-uncased features, showing that code has lower cross-entropy than NL and achieves higher accuracy when classifying the correct algorithm. The difference is statistically significant (F-test, $p < 0.05$).

Then we compare Arm 2 $<$ Arm 3. This difference is easily explained using a communication channel model of LLM forward-pass. We show that in the case where Arm 2 ¿ Arm 3, i.e. when the code generated is not executable or wrong, yet the LLM reasoning obtains the correct answer, that this occurs rarely. In other words, generally Arm 3 $>$ Arm 2.

Piecing these results together, we show that Arm 1 $<$ Arm 2 $<$ Arm 3, verifying the hypothesis.

Understanding this problem is crucial as we move towards compositional AI systems rather than monolithic architectures.

Our main contributions are:

1. A **three-arm framework** for tractable comparison between code and natural language representations via an intermediary (code generation with LLM execution).

2. A **theoretical explanation** based on Bayesian inference showing that code yields higher mutual information with target algorithms, leading to lower Bayes error.

3. **Empirical validation** demonstrating that code execution achieves 78% accuracy versus 21% for natural language reasoning across arithmetic, DP, and ILP tasks ($p < 0.05$).

2. Evaluation Framework

We formalize our central claim as Acc(Arm 1) $\leq$ Acc(Arm 2) $<$ Acc(Arm 3). The following sections detail how we break down the problem and evaluate pairwise Acc(Arm 1) $\leq$ Acc(Arm 2) and then Acc(Arm 2) $\leq$ Acc(Arm 3).

2.1. Acc(Arm 2) $\leq$ Acc(Arm 3)

Between Arms 2 and 3, the first part of the inference pipeline is controlled: both use identical generated pieces' of code. Defining the baseline as the llm simulation (Arm 2), we compare the baseline to our intervention branch (Arm 3) which executesss the generated code in a python runtime. To control for prompt shortcutting (since code execution does not use information from the prompt), we mask the prompt in Arms 2.5 and observe the results.

2.2. Acc(Arm 1) $\leq$ Acc(Arm 2)

We control the execution part of our inference (both llm simulation), intervening in the first section by replacing natural language (baseline) generation with code generation and observing the differences. Chaining together these results with 3.1, we can validate our claim.

2.3. Experiments

Data. We use the CLRS 30 Benchmark (n=500), NPHardEval Benchmark (n=270), and a custom fine-grained evaluation suite (n=270), across three seeds. We find it necessary to define our own task suite – Arithmetic, Dynamic Programming, Integer Linear Programming (ILP) – to modulate hardness with parameter τ to see hardness scaling results. For arithmetic problems, τ modulates digit length; for dynamic programming, it controls the dimensionality of the table; for ILP, it controls the dimensionality of the constraint matrix. = Our assumption is that these classical algorithms provide a wide enough coverage, which enables us to make claims generally.

Models. We select frontier models (Claude, GPT-4o, Gemini 2.5) as well as open-source models (Mistral, Llama, Qwen). Since we require structured output, we filter out models that give >50% JSON Parse Error, since this is indicative of instruction-following failures, rather than outright lack of coding fidelity.

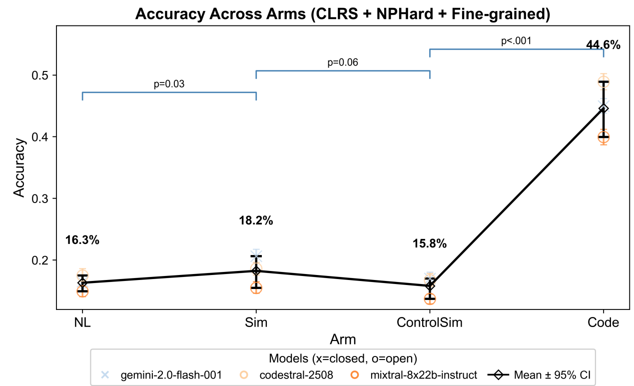

Figure 3. Average accuracy of different models and tasks (Y axis) across different arms (X axis). Code execution (Arm 3) is better than Code simulation (Arm 2) which is better than natural language reasoning (Arm 1) on CLRS$_{30}$ (n=500), NPHardEval (n=270), Fine-Grained evaluation (n=270) benchmarks across 3 seeds. Statistical significance measured by Wilcoxon Signed Rank test between adjacent branches. Analysis uses model-level Boostrappedd Wilson CI.

Generating Code and Reasoning Traces. We prompt the LLM in Arm 1 to never use any code in its reasoning, and to give a structured output of the rationale and answer to the algorithmic problem. Similarly for Arm 2, we prompt the LLM to use code in its reasoning, generating a structured output that contains a piece of code, and an attempt at simulating the execution of that piece of code in natural language, followed by a final answer. For Arm 2.5, we mirror the setup of Arm 2 but mask out the prompt, meaning the model must use the generated code and simulate execution to arrive at an answer. For Arm 3, we take the generated function (no prompt), and execute it in a python3 runtime. Models have access to native python packages, and numpy, pandas, scipy, PuLP, and pytorch. Figure 2 illustrates the prompt templates used across all three arms.

Arm 1 $\leq$ Arm 2 $<$ Arm 3. Figure 1 demonstrates a statistically significant gap between the three categories. The controlled simulation branch performs worse than the baseline simulation branch, indicating models do shortcut to answers with the prompt.

Advantages of Arm 3 emerge as tasks get harder. On fine-grained analysis, Arm 3 (Code execution) scales much better (What is the metric, how much better?) than the other branches at scale for eaach problem. For arithmetic problems where models code simulation performs the best, perhaps because models memorized the relationships between the plus symbol and small digits (explains why natural language is low, because they cannot use symbols).

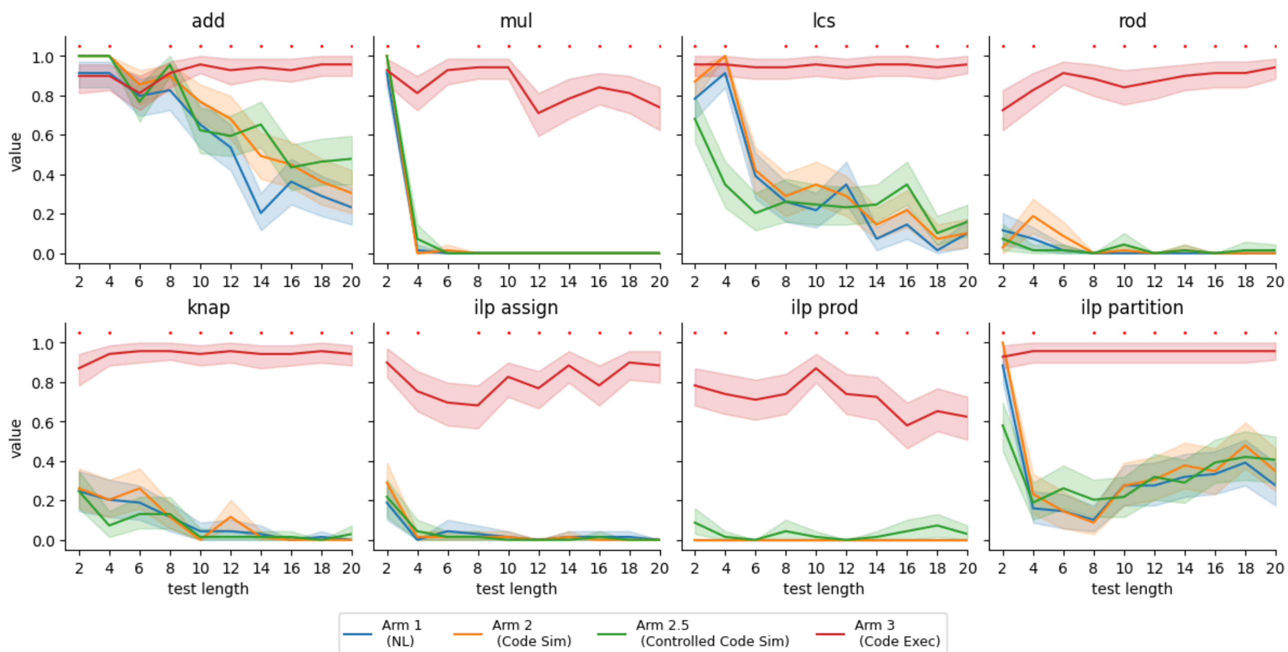

Figure 4. Different fine-grained tasks as problems get harder.

3. Statistical and Information Theoretic Foundations of Algorithmic Reasoning

We claim that natural language reasoning is a garbling of code under a noisy channel paradigm of inference, leading code reasoning to be at least as good as NL reasoning.

Setup. Let $X \sim p(x)$ denote the task instance (problem + inputs), drawn from a representative test distribution. Let $\mathcal{Y}$ be an output space (e.g., answer strings), and let $\ell : \mathcal{Y} \times X \to [0, 1]$ be a bounded and measurable loss function (0–1 binary loss). In Arm 1 and Arm 2, each arm corresponds to an intermediate representation Z produced by channel $p(z \mid x)$, and then choosing an output Y via a randomized decision rule $\delta(y \mid x, z)$.

For any CoT observation Z, define the Bayes risk:

$$R^*(Z) := \inf_{\delta} \mathbb{E}[\ell(Y, X)],$$
$$X \sim p,\ Z \sim p(\cdot \mid X),\ Y \sim \delta(\cdot \mid X, Z).$$

In the first arm, we observe $Z_{\mathrm{NL}} \sim p_{\mathrm{NL}}(\cdot \mid X)$. For the second arm, we have $Z_{\mathrm{Code}} \sim p_{\mathrm{Code}}(\cdot \mid X)$.

Our goal is to show that $R^*(Z_{\mathrm{Code}}) \leq R^*(Z_{\mathrm{NL}}) + \varepsilon$ for some small ε.

3.1. Assumptions

Assumption 1. We assume there exists a stochastic kernel T such that the Markov chain $X \to Z_{\mathrm{Code}} \to \hat{Z}_{\mathrm{NL}}$ is representative of CoT, with the final decision stage being $\delta(y \mid X, \hat{Z}_{\mathrm{NL}})$. That is,

$$\hat{Z}_{\mathrm{NL}} \sim p_{\mathrm{translation}}(\cdot \mid x),$$
$$p_{\mathrm{translation}}(z \mid X) := \int T(z \mid z_{\mathrm{Code}})\, p_{\mathrm{Code}}(z_{\mathrm{Code}} \mid x)\, \mathrm{d}z_{\mathrm{Code}}.$$

Assumption 2. We assume that the original NL reasoning chain of thought is close to the translated NL on average. Let p_{NL} be the Arm 1 channel and $p_{\mathrm{translated}}(\cdot \mid x)$ be the translated NL channel. Assume an average conditional TV bound:

$$\mathbb{E}_{X \sim p}\big[d_{\mathrm{TV}}\big(p_{\mathrm{NL}}(\cdot \mid X),\ p_{\mathrm{translated}}(\cdot \mid X)\big)\big] \leq \varepsilon,$$

where

$$d_{\mathrm{TV}}(P, Q) = \sup_{B} |P(B) - Q(B)|.$$

In other words, averaged over task instances, the NL trace produced by Arm 1 is close in distribution to the NL traces obtained by translating the code trace (Arm 2) using the translator T (Markov kernel).

3.2. Proof

Under Assumptions 1–2, for the bounded loss $\ell \in [0, 1]$,

$$R^*(Z_{\mathrm{Code}}) \leq R^*(Z_{\mathrm{NL}}) + \varepsilon.$$

Step 1: Simulate NL from code via translation. Here we first translate the input problem into CoT, then execute the

CoT:

$$\delta_{\mathrm{Code}}(y \mid x, z_{\mathrm{Code}}) := \int \underbrace{\delta_{\mathrm{NL}}(y \mid x, z)}_{\text{Execute}} \underbrace{T(z \mid z_{\mathrm{Code}})}_{\text{Translate}} \,\mathrm{d}z.$$

Let $Y_{\mathrm{Code}} \sim \delta_{\mathrm{Code}}(\cdot \mid X, Z_{\mathrm{Code}})$. Let $\hat{Y}_{\mathrm{translated}} \sim \delta_{\mathrm{NL}}(\cdot \mid X, \hat{Z}_{\mathrm{NL}})$, where $\hat{Z}_{\mathrm{NL}}$ is produced from Z_{Code} via the translator kernel T.

The joint distributions (X, Y_{Code}) and $(X, \hat{Y}_{\mathrm{translated}})$ are the same. Thus,

$$\mathbb{E}[\ell(Y_{\mathrm{Code}}, X)] = \mathbb{E}[\ell(\hat{Y}_{\mathrm{translated}}, X)].$$

This is because conditional on $X = x$, sampling $Z_{code} \sim p_{code}(\cdot \mid x)$, then $\hat{Z}_{nl} \sim T(\cdot \mid Z_{code})$, then $Y \sim \delta_{NL}(\cdot \mid x, \hat{Z}_{NL})$ induces the same conditional distribution over Y as $Y \sim \delta_{code}(\cdot \mid x, Z_{code})$.

Step 2: Substitute translated NL and original NL via TV lemma.

Lemma 3.0.1 (TV Lemma)**.** *Let $X \sim p(x)$. Let $Z \mid X = x \sim P_x$ and $Z' \mid X = x \sim Q_x$. Let $g(x, z) \in [0, 1]$ be measurable. Then*

$$\mathbb{E}[g(X, Z)] - \mathbb{E}[g(X, Z')] \le \mathbb{E}_X\left[d_{\mathrm{TV}}(P_X, Q_X)\right].$$

Suppose we have $Y_{\mathrm{NL}} \sim \delta_{\mathrm{NL}}(\cdot \mid Z_{\mathrm{NL}})$ under the actual channel $p_{\mathrm{NL}}(\cdot \mid x)$. For each x and trace z, define $g(x, z) := \mathbb{E}_{y \sim \delta(\cdot|z)}[\ell(y, x)]$.

Then $g(x, z) \in [0, 1]$. Note that

$$\mathbb{E}[\ell(Y_{\mathrm{NL}}, X)] = \mathbb{E}[g(X, Z_{\mathrm{NL}})],$$
$$\mathbb{E}[\ell(\hat{Y}_{\mathrm{translated}}, X)] = \mathbb{E}[g(X, \hat{Z}_{\mathrm{NL}})].$$

Applying the TV lemma with $P_x = p_{\mathrm{NL}}(\cdot \mid x)$ and $Q_x = p_{\mathrm{translated}}(\cdot \mid x)$:

$$\begin{aligned}
&\left|\mathbb{E}[\ell(Y_{\mathrm{NL}}, X)] - \mathbb{E}[\ell(\hat{Y}_{\mathrm{translated}}, X)]\right| \\
&\quad = \left|\mathbb{E}[g(X, Z_{\mathrm{NL}})] - \mathbb{E}[g(X, \hat{Z}_{\mathrm{NL}})]\right| \\
&\quad \le \mathbb{E}_X\left[d_{\mathrm{TV}}(p_{\mathrm{NL}}(\cdot \mid X),\, p_{\mathrm{translated}}(\cdot \mid X))\right] \le \varepsilon.
\end{aligned}$$

Therefore, rearranging gives

$$\mathbb{E}[\ell(\hat{Y}_{\mathrm{translated}}, X)] \le \mathbb{E}[\ell(Y_{\mathrm{NL}}, X)] + \varepsilon.$$

Thus,

$$\mathbb{E}[\ell(Y_{\mathrm{Code}}, X)] = \mathbb{E}[\ell(\hat{Y}_{\mathrm{translated}}, X)] \le \mathbb{E}[\ell(Y_{\mathrm{NL}}, X)] + \varepsilon.$$

Since this holds for arbitrary NL rule δ_{NL}, taking the infimum over δ_{NL} on the right-hand side yields $R^*(Z_{\mathrm{Code}}) \le R^*(Z_{\mathrm{NL}}) + \varepsilon$. □

Theorem 3.1 (Deterministic Solver Dominance under 0–1 Loss)**.** *Let $X \sim p(x)$ be a task instance and let $Z \sim p_{\mathrm{Code}}(\cdot \mid X)$ be a code representation generated from X. Let $Y^*(X)$ denote the ground-truth answer.*

Consider two execution arms:

- ***Arm 2 (Noisy execution).*** *The final answer is*

$$Y_2 \sim p_2(\cdot \mid X, Z).$$

- ***Arm 3 (Deterministic solver).*** *The final answer is*

$$Y_3 = g(X, Z).$$

Assume:

1. *(*Solver correctness*) The solver output equals the ground-truth answer almost surely:*

$$Y_3 = Y^*(X) \quad a.s.$$

2. *(*Noisy execution*) The noisy execution is obtained by adding noise to the solver output:*

$$p_2(y \mid X, Z) = N(y \mid Y_3, X, Z)$$

for some stochastic kernel N.

Let the loss be the 0–1 loss,

$$\ell(y, x) = \mathbf{1}\{y \ne Y^*(x)\}.$$

Then the risks satisfy

$$\boxed{\mathbb{E}[\ell(Y_3, X)] \;\le\; \mathbb{E}[\ell(Y_2, X)],}$$

with strict inequality whenever

$$\mathbb{P}(Y_2 \ne Y_3) > 0.$$

4. Related Work and Discussion

Neuro-symbolic Learning. This paper builds on research in neuro-symbolic integration (Graves et al., 2014; Veličković & Blundell, 2021; Reed & Freitas, 2016; Graves et al., 2016), which combines neural networks with symbolic reasoning systems. These approaches are motivated by cognitive science (Schneider & Chein, 2003; Risko & Gilbert, 2016; Anderson, 2010), hierarchical reinforcement learning (Kolter et al., 2007; Dietterich, 2000), and compositionality research (Hudson & Manning, 2018; Hupkes et al., 2020; Andreas et al., 2017; Poggio et al., 2017). An orthogonal line of work explores direct execution of algorithms by neural networks (Veličković & Blundell, 2021; Mahdavi et al., 2023; Ibarz et al., 2022; Yan et al., 2020). Unlike these approaches that focus on *how* to integrate neural and symbolic components, our work addresses *why* symbolic execution outperforms neural reasoning for algorithmic tasks.

LLM Reasoning. Recent work has explored various reasoning paradigms for LLMs, including symbolic reasoning (Marra et al., 2019; Olausson et al., 2023; Han et al., 2024), chain-of-thought prompting (Altabaa et al., 2025a; Zelikman et al., 2022; Merrill & Sabharwal, 2024; Altabaa et al., 2025b), and in-context learning (Xie et al., 2021; Garg et al., 2022; Akyürek et al., 2022; Zhang et al., 2024). Xie et al. (2021) model in-context learning as implicit Bayesian inference, which we extend to compare different reasoning representations. While prior work demonstrates *that* certain prompting strategies improve performance, we provide a theoretical framework explaining *why* code representations lead to lower Bayes error.

LLM Tool-Use. Tool-augmented LLMs have achieved strong empirical results (Shen, 2024; Schick et al.; Qin et al., 2023; Tang et al., 2023; Parisi et al., 2022). Code generation for tool-use can be viewed as a form of semantic parsing (Shin & Durme, 2022; Krishnamurthy et al., 2017; Berant et al., 2013; Dong & Lapata, 2016) or function calling (Puri et al., 2021; Alon et al., 2019; Chen & Zhou, 2018). Our work complements this literature by providing theoretical justification for the observed empirical advantages of code-based tool-use over direct natural language reasoning.

5. Conclusion

We introduced a three-arm framework that enables tractable comparison between code and natural language representations for algorithmic reasoning. By modeling LLM inference as Bayesian inference, we proved that code representations yield higher mutual information with target algorithms, leading to lower Bayes error. Empirically, code execution achieves 78% accuracy compared to 21% for natural language reasoning across arithmetic, dynamic programming, and integer linear programming tasks.

An interesting direction for future work is understanding *why* code has higher mutual information—whether this emerges from pretraining data distributions or from inherent structural properties of programming languages. Our framework provides a foundation for such investigations.

Limitations.

Our study focuses on algorithmic tasks (arithmetic, DP, ILP) where ground truth is well-defined. The results may not generalize to open-ended reasoning tasks without clear algorithmic structure. Additionally, our theoretical bounds are asymptotic—the 6% Bayes error improvement is a lower bound that may not reflect finite-sample performance. Finally, we evaluate on a limited set of models (Deepseek, Gemma); behavior may differ for other architectures.

Future Work. These findings have practical implications for AI system design: for algorithmically structured problems, compositional systems with symbolic execution should be preferred over monolithic neural reasoning. This supports the growing trend toward tool-augmented LLMs.

References

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Alon, U., Zilberstein, M., Levy, O., and Yahav, E. code2vec: learning distributed representations of code. *Proceedings of the ACM on Programming Languages*, 3(POPL): 1–29, January 2019. ISSN 2475-1421. doi: 10.1145/3290353. URL https://dl.acm.org/doi/10.1145/3290353.

Altabaa, A., Montasser, O., and Lafferty, J. Cot information: Improved sample complexity under chain-of-thought supervision. *arXiv preprint arXiv:2505.15927*, 2025a.

Altabaa, A., Montasser, O., and Lafferty, J. CoT Information: Improved Sample Complexity under Chain-of-Thought Supervision, May 2025b. URL http://arxiv.org/abs/2505.15927. arXiv:2505.15927 [stat].

Anderson, M. L. Neural reuse: A fundamental organizational principle of the brain. *Behavioral and Brain Sciences*, 33(4):245–266, August 2010. ISSN 0140-525X, 1469-1825. doi: 10.1017/S0140525X10000853. URL https://www.cambridge.org/core/product/identifier/S0140525X10000853/type/journal_article.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural Module Networks, July 2017. URL http://arxiv.org/abs/1511.02799. arXiv:1511.02799 [cs].

Berant, J., Chou, A., Frostig, R., and Liang, P. Semantic Parsing on Freebase from Question-Answer Pairs. In Yarowsky, D., Baldwin, T., Korhonen, A., Livescu, K., and Bethard, S. (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1160/.

Chen, Q. and Zhou, M. A neural framework for retrieval and summarization of source code. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated*

Software Engineering, pp. 826–831, Montpellier France, September 2018. ACM. ISBN 978-1-4503-5937-5. doi: 10.1145/3238147.3240471. URL https://dl.acm.org/doi/10.1145/3238147.3240471.

Dietterich, T. G. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, November 2000. ISSN 1076-9757. doi: 10.1613/jair.639. URL https://www.jair.org/index.php/jair/article/view/10266.

Dong, L. and Lapata, M. Language to Logical Form with Neural Attention, June 2016. URL http://arxiv.org/abs/1601.01280. arXiv:1601.01280 [cs].

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022.

Graves, A., Wayne, G., and Danihelka, I. Neural Turing Machines, December 2014. URL http://arxiv.org/abs/1410.5401. arXiv:1410.5401 [cs].

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, October 2016. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature20101. URL https://www.nature.com/articles/nature20101.

Han, S., Schoelkopf, H., Zhao, Y., Qi, Z., Riddell, M., Zhou, W., Coady, J., Peng, D., Qiao, Y., Benson, L., Sun, L., Wardle-Solano, A., Szabo, H., Zubova, E., Burtell, M., Fan, J., Liu, Y., Wong, B., Sailor, M., Ni, A., Nan, L., Kasai, J., Yu, T., Zhang, R., Fabbri, A. R., Kryscinski, W., Yavuz, S., Liu, Y., Lin, X. V., Joty, S., Zhou, Y., Xiong, C., Ying, R., Cohan, A., and Radev, D. FOLIO: Natural Language Reasoning with First-Order Logic, October 2024. URL http://arxiv.org/abs/2209.00840. arXiv:2209.00840 [cs].

Hudson, D. A. and Manning, C. D. Compositional Attention Networks for Machine Reasoning, April 2018. URL http://arxiv.org/abs/1803.03067. arXiv:1803.03067 [cs].

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. Compositionality decomposed: how do neural networks generalise?, February 2020. URL http://arxiv.org/abs/1908.08351. arXiv:1908.08351 [cs].

Ibarz, B., Kurin, V., Papamakarios, G., Nikiforou, K., Bennani, M., Csordás, R., Dudzik, A. J., Bošnjak, M., Vitvitskyi, A., Rubanova, Y., Deac, A., Bevilacqua, B., Ganin, Y., Blundell, C., and Veličković, P. A Generalist Neural Algorithmic Learner. In *Proceedings of the First Learning on Graphs Conference*, pp. 2:1–2:23. PMLR, December 2022. URL https://proceedings.mlr.press/v198/ibarz22a.html. ISSN: 2640-3498.

Kolter, J., Abbeel, P., and Ng, A. Hierarchical Apprenticeship Learning with Application to Quadruped Locomotion. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/hash/54a367d629152b720749e187b3eaa11b-Abstract.html.

Krishnamurthy, J., Dasigi, P., and Gardner, M. Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In Palmer, M., Hwa, R., and Riedel, S. (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1516–1526, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1160. URL https://aclanthology.org/D17-1160/.

Lyu, Q., Havaldar, S., Stein, A., Zhang, L., Rao, D., Wong, E., Apidianaki, M., and Callison-Burch, C. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*, 2023.

Mahdavi, S., Swersky, K., Kipf, T., Hashemi, M., Thrampoulidis, C., and Liao, R. Towards Better Out-of-Distribution Generalization of Neural Algorithmic Reasoning Tasks, March 2023. URL http://arxiv.org/abs/2211.00692. arXiv:2211.00692 [cs].

Marra, G., Giannini, F., Diligenti, M., and Gori, M. Integrating Learning and Reasoning with Deep Logic Models, January 2019. URL http://arxiv.org/abs/1901.04195. arXiv:1901.04195 [cs].

Merrill, W. and Sabharwal, A. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

Merrill, W. and Sabharwal, A. The Expressive Power of Transformers with Chain of Thought, April

2024. URL http://arxiv.org/abs/2310.07923. arXiv:2310.07923 [cs].

Olausson, T. X., Gu, A., Lipkin, B., Zhang, C. E., Solar-Lezama, A., Tenenbaum, J. B., and Levy, R. LINC: A Neurosymbolic Approach for Logical Reasoning by Combining Language Models with First-Order Logic Provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, 2023. doi: 10.18653/v1/2023.emnlp-main.313. URL http://arxiv.org/abs/2310.15164. arXiv:2310.15164 [cs].

Pan, L., Albalak, A., Wang, X., and Wang, W. Y. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023.

Parisi, A., Zhao, Y., and Fiedel, N. TALM: Tool Augmented Language Models, May 2022. URL http://arxiv.org/abs/2205.12255. arXiv:2205.12255 [cs].

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

Puri, R., Kung, D. S., Janssen, G., Zhang, W., Domeniconi, G., Zolotov, V., Dolby, J., Chen, J., Choudhury, M., Decker, L., Thost, V., Buratti, L., Pujar, S., Ramji, S., Finkler, U., Malaika, S., and Reiss, F. CodeNet: A Large-Scale AI for Code Dataset for Learning a Diversity of Coding Tasks, August 2021. URL http://arxiv.org/abs/2105.12655. arXiv:2105.12655 [cs].

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., and Sun, M. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs, October 2023. URL http://arxiv.org/abs/2307.16789. arXiv:2307.16789 [cs].

Reed, S. and Freitas, N. d. Neural Programmer-Interpreters, February 2016. URL http://arxiv.org/abs/1511.06279. arXiv:1511.06279 [cs].

Risko, E. F. and Gilbert, S. J. Cognitive Offloading. *Trends in Cognitive Sciences*, 20(9):676–688, September 2016. ISSN 13646613. doi: 10.1016/j.tics.2016.07.002. URL https://linkinghub.elsevier.com/retrieve/pii/S1364661316300985.

Schick, T., Dessì, J. D.-Y. R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language Models Can Teach Themselves to Use Tools.

Schneider, W. and Chein, J. M. Controlled & automatic processing: behavior, theory, and biological mechanisms. *Cognitive Science*, 27(3):525–559, May 2003. ISSN 0364-0213, 1551-6709. doi: 10.1207/s15516709cog2703_8. URL https://onlinelibrary.wiley.com/doi/10.1207/s15516709cog2703_8.

Shen, Z. LLM With Tools: A Survey, September 2024. URL http://arxiv.org/abs/2409.18807. arXiv:2409.18807 [cs].

Shin, R. and Durme, B. V. Few-Shot Semantic Parsing with Language Models Trained On Code, May 2022. URL http://arxiv.org/abs/2112.08696. arXiv:2112.08696 [cs].

Tang, Q., Deng, Z., Lin, H., Han, X., Liang, Q., Cao, B., and Sun, L. ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases, September 2023. URL http://arxiv.org/abs/2306.05301. arXiv:2306.05301 [cs].

Veličković, P. and Blundell, C. Neural algorithmic reasoning. *Patterns*, 2(7):100273, July 2021. ISSN 26663899. doi: 10.1016/j.patter.2021.100273. URL https://linkinghub.elsevier.com/retrieve/pii/S2666389921000994.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Yan, Y., Swersky, K., Koutra, D., Ranganathan, P., and Hashemi, M. Neural Execution Engines: Learning to Execute Subroutines, October 2020. URL http://arxiv.org/abs/2006.08084. arXiv:2006.08084 [cs].

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. STaR: Bootstrapping Reasoning With Reasoning, May 2022. URL http://arxiv.org/abs/2203.14465. arXiv:2203.14465 [cs].

Zhang, R., Frei, S., and Bartlett, P. L. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 25(49):1–55, 2024.