RELIC: Evaluating Compositional Instruction Following via Language Recognition

Anonymous Author(s)
Affiliation
Address
email

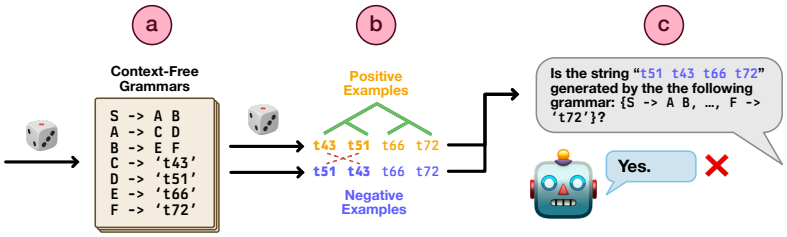

Figure 1: RELIC is an evaluation framework for *compositional instruction following*, where we **(a)** stochastically generate context-free grammars (e.g., sets of instructions) of given complexities, **(b)** sample positive and negative strings (e.g., tasks) for each grammar, and **(c)** prompt models to classify whether the strings are generated by those grammars.

Abstract

Large language models (LLMs) are increasingly expected to perform tasks based only on a specification of the task provided in context, without examples of inputs and outputs; this ability is referred to as instruction following. To evaluate models' ability to perform a task provided in context, we introduce the Recognition of Languages In-Context (RELIC) framework, where the task is to determine if a string is generated by a context-free grammar. This requires composing together a large number of instructions (grammar productions) retrieved from the context. Because the languages are synthetic, the task can be increased in complexity as LLMs' skills improve in the future, and new instances can be automatically generated, mitigating data contamination concerns. We evaluate state-of-the-art LLMs on RELIC and find that their accuracy can be reliably predicted from the complexity of the grammar and the individual example strings, and that even the most advanced LLMs currently available show near-chance performance on more complex grammars and samples, in line with theoretical expectations. We also analyze how LLMs attempt to solve increasingly difficult reasoning tasks, and find that as the complexity of the language recognition task increases, models switch from following complex instructions to relying on shallow heuristics.

1 Introduction

Large language models (LLMs) are increasingly expected to solve tasks "zero-shot," using only a specification of the task provided in the LLM's context window, without examples of inputs and outputs or any additional training. These instructions are often *compositional*, involving the interaction of multiple rules or constraints. Even a simple question such as "using my credit card statement, tell

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

me if I spent more money in 2025 than I did on average in the past three years" (inspired by Zhou et al. 2024) requires an LLM to retrieve a large number of items from its context and perform multiple operations in a particular sequence (in this example, summing and averaging them and comparing the outputs). As the complexity of such instructions increases, the system needs to use its context in a sophisticated way to identify the relevant instructions and combine them appropriately. The importance of compositional instruction following highlights the need for a reliable benchmark with controllable complexity that can measure LLMs' abilities in this area as they improve in the future.

We introduce the Recognition of Languages In-Context (RELIC) framework as a means of studying how well LLMs can understand and follow complex compositional instructions provided in prompts. Here, an model is tasked with solving the *language recognition* task: is an arbitrary string derivable from the rules of a given grammar? Earlier work has explored instruction following through the lens of the recognition of regular languages by models fine-tuned for the task (e.g., Finlayson et al. 2022; see appendix A for a full review of related work). Here, we extend this paradigm in both complexity and flexibility to ask: can LLMs perform the language-recognition task for the substantially more complex class of context-free languages, and do so in an in-context setting, where they are not provided any training on the task or the particular grammars? We build RELIC as a data generating process, which stochastically samples new grammars of desired complexity and produces unlimited amounts of new evaluation examples for these grammars. This property of the benchmark addresses issues of data contamination and benchmark saturation which plague static benchmarks.

Using this framework, we generate and release a first static benchmark, RELIC-500, which consists of 200 grammars which vary in complexity (up to 500 nonterminal production rules), and sets of positive and negative examples for each grammar (of lengths up to 50 symbols). We evaluate a range of frontier LLMs on RELIC-500. Most of the models performed substantially above chance on simpler grammars, indicating an understanding of the general structure of the task; but all models, including OpenAI's most advanced reasoning model `o3`, had near-chance accuracy on the larger grammars, which are still orders of magnitude smaller than the grammars needed to define commonly-used programming languages or approximate human languages. In a qualitative analysis of the chain-of-thought tokens that LLMs produce before providing their response, we find that in many cases models rely on incorrect heuristics; quantitative analysis of these tokens shows that the amount of tokens does not grow as would be expected from the correct strategy to perform the task.

2 RELIC: Recognizing Languages In-Context

Following Clark (2017, 2018), we stochastically generate grammars which are parameterized by four values: the number of terminal symbols n_{term}, non-terminal symbols n_{nonterm}, lexical production rules n_{lex}, and non-lexical production rules n_{nonlex}. We start by defining n_{term} symbols $\Sigma = \{\texttt{t}_1, \texttt{t}_2, \ldots\}$; and n_{nonterm} symbols $V = \{\texttt{NT}_1, \texttt{NT}_2, \ldots\}$. We then sample n_{lex} lexical production rules $\texttt{NT}_a \rightarrow$ '$\texttt{t}_b$' from the set of pairs $V \times \Sigma$, and we sample n_{nonterm} non-lexical production rules $\texttt{NT}_a \rightarrow \texttt{NT}_b\ \texttt{NT}_c$ from the set of all triples $(\{\texttt{S}\} \cup V) \times V \times V$, where S is a privileged start symbol. The grammar is then trimmed to remove any non-lexical rules which do not lead to a lexical rule, and any lexical rules which are inaccessible from non-lexical productions. The result is a coherent context-free grammar with at most as many terminals, nonterminals, lexical productions, and non-lexical productions as the generating parameters.

We generate a first static benchmark, which we refer to as RELIC-500. We sample 200 grammars where all parameters (n_{term}, n_{nonterm}, n_{lex} and n_{nonlex}) are less than 500. For each grammar, we sample up to 10 positive and negative strings for each length $1 \leq \ell \leq 50$. We sample positive strings by treating each grammar as a probabilistic context-free grammar with uniform probability assigned to production rules sharing a left-hand symbol; and negative strings from a unigram model over the terminal symbols Σ^+, rejecting any which parse.

Intended use. RELIC is designed as a *zero-shot* evaluation of a model's ability to use instructions provided in-context *without providing positive and negative exemplars*, and without fine-tuning on this task. Evaluating models in an in-context few-shot setting could lead to higher accuracy, but any increases in accuracy could be due to heuristics that distinguish the two classes with some accuracy but fail to strictly test the model's ability to follow in-context instructions. We release RELIC-500 and the codebase to generate new grammars and examples at the following anonymized GitHub repository: https://anonymous.4open.science/r/relic-C363/.

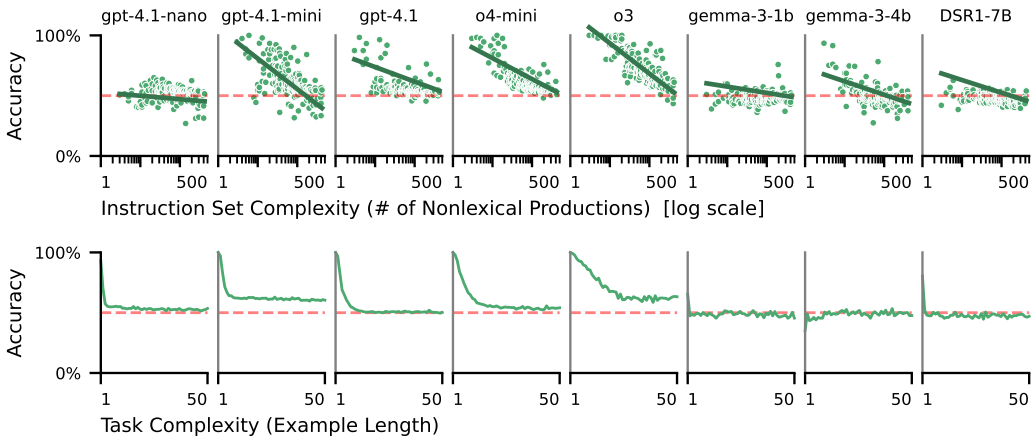

Figure 2: Accuracy on RELIC-500 reduces to near-chance (dashed lines) for all models as instruction set complexity (number of nonlexical productions in the grammar, **top row**) and task complexity (example length, **bottom row**) increase.

3 Evaluating Frontier LLMs on RELIC-500

We evaluate eight LLMs on RELIC-500. First, three models from OpenAI's GPT line that were the newest at the time of writing: `gpt-4.1-nano`, `gpt-4.1-mini`, and `gpt-4.1` (released on April 14; OpenAI 2025a). Second, we evaluate two "reasoning" models from OpenAI, `o4-mini` and `o3` (OpenAI, 2025b). All evaluations of OpenAI's API-based models were carried out between 14 April 2025 and 1 May 2025. Third, we evaluate two models from Google's Gemma 3 family of open-weights instruction-tuned language models (`gemma-3-1b-it` and `gemma-3-4b-it`; Gemma Team et al. 2025). Finally, we evaluate an open-weights reasoning model, `DeepSeek-R1-Distill-Qwen-7B` (DeepSeek-AI et al., 2025). See appendix E for more details on API costs, compute, and evaluation hyperparameters.

For each grammar G and each string s, we prompt the models with a description of the language recognition task, the grammar G, and the string s, and ask the model to classify the string according to whether or not it is generated by the grammar (see appendix B for an example prompt). We use a regular expression over the output to extract whether the model classified the example as positive or negative; we classify the response as "unknown" if the model fails to offer a prediction. We evaluate `gpt-4.1-nano`, `gpt-4.1-mini`, `gpt-4.1`, and `o4-mini` on the full RELIC-500 dataset; for `o3`, `gemma-3-1b-it`, `gemma-3-4b-it`, and `DeepSeek-R1-Distill-Qwen-7B`, we subsample each grammar's data to have at most two examples per length per type due to the increased cost of running evaluations on these models.

3.1 Performance Decreases as Grammar and Example Complexity Increases

When grammars are small and example strings are short, most models display near-perfect accuracy on RELIC as shown in fig. 2, indicating that models exhibit the capacity to solve instances of the language recognition task in principle. However, for all models, performance on RELIC-500 decreases as a function of a grammar's complexity, as quantified by each of the four grammar parameters. Among these parameters, the number of non-lexical productions n_{nonlex} is most strongly anti-correlated with performance. On a per-model basis, we observe a roughly log-linear relationship between the number of non-lexical productions and performance: though some models have high accuracy on small grammars, as n_{nonlex} approaches 500 all models are at or below chance performance (fig. 2, top).

Model performance is also affected by the complexity of individual examples. As example length ℓ increases, models' mean accuracy over both positive and negative examples decreases (fig. 2, bottom). This drop-off happens quite rapidly, with an inflection point occurring at between $\ell = 5$ and $\ell = 15$ depending on the model. A regression shows that the effects of $\log(n_{\text{nonlex}})$ and $\log(\ell)$ are highly significant (see tables 1 and 2 in appendix C).

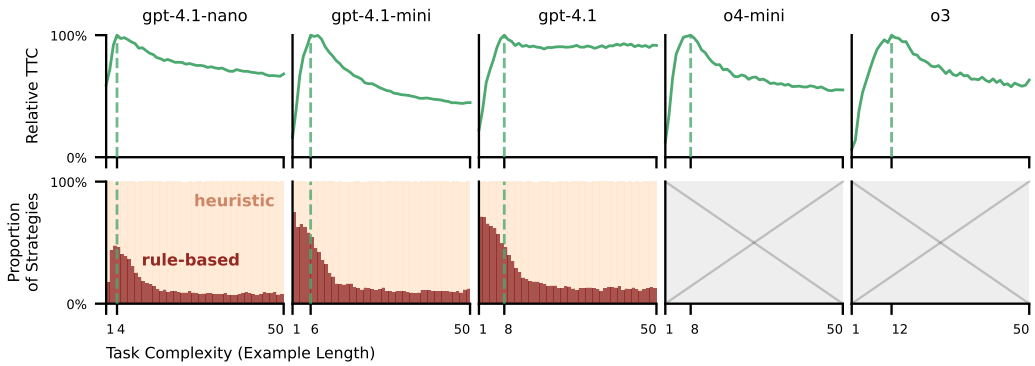

Figure 3: As task complexity (example length) increases, the test-time compute (TTC) expended by models peaks early and then diminishes (**top row**; TTC is computed as the mean number of completion tokens produced for examples of a given length, relative to the length for which the mean number of tokens is highest). Concomitantly, models shift from pursuing rule-based approaches to relying on heuristic strategies (**bottom row**). `o4-mini` and `o3` do not provide full CoTs, so we cannot classify these models' strategies.

3.2 Models Don't Use Chains-of-Thought Effectively

Inn-context recognition for CFGs is known to be in the complexity classes AC^1 and NC^1-hard (Ruzzo, 1980; Venkateswaran, 1991; Greenlaw et al., 1991). In contrast, transformers without chain-of-thought are in the complexity class TC^0 (Merrill and Sabharwal, 2023). Thus, under standard complexity conjectures, solving RELIC by implementing parsing should require the model to spend more test-time compute (TTC) on longer inputs. Standard context-free parsing algorithms run in time $\Theta(n^3)$, and it is unlikely that this can be significantly improved (Lee, 2002). Applying results relating TTC to algorithmic runtime (Merrill and Sabharwal, 2024), a transformer can solve RELIC only by scaling test-time-compute superlinearly in the length of the input string. By contrast, fig. 3 shows that the number of tokens generated *diminishes* for longer input strings, suggesting models are backing off to heuristics rather than recognizing long strings by parsing them. These changes motivate LLM systems which are either not so limited in their inference compute or which can solve the problem in a more compute-efficient manner.

3.3 Models Shift to Heuristics as Complexity Increases

Qualitative analysis of model completions reveals divergent strategies models employ to solve the task. In some cases, models attempt to solve the task by algorithmic means, either by building a parse table for a string or by exhaustively searching the production rules for a licit derivation; in other cases, models rely on heuristic arguments for why a string 'seems' (un)likely to be derivable from the provided grammar (see appendix F for example completions). To gauge strategy use, we use `o4-mini` as an LLM-judge (Zheng et al., 2023) to classify the responses of the three best-performing models reporting full completions: the `gpt-4.1` family. We classify responses into 'rule-based' or 'heuristic' strategies irrespective of final correctness. Figure 3 shows that as task complexity increases, models shift from pursuing rule-based approaches to relying on shallow heuristic arguments. This shift is concomitant with the decrease in both overall accuracy and relative test-time-compute expended by the models on solving the task.

4 Discussion

We introduce RELIC, a method for studying how well language models can follow complex compositional instructions through the lens of in-context language recognition. We show that current frontier LLMs struggle substantially on this task, suggesting substantial room for improvement in generalized instruction following. We diagnose how and why models fail by comparing their compute expenditure to theoretical predictions, finding that, as inputs become complex, models shift to relying on unsound heuristics.

References

E. Akyürek, B. Wang, Y. Kim, and J. Andreas. In-context language learning: Architectures and algorithms, 23 Jan. 2024. URL http://arxiv.org/abs/2401.12973.

S. Bhattamishra, K. Ahuja, and N. Goyal. On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Stroudsburg, PA, USA, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.576. URL http://dx.doi.org/10.18653/v1/2020.emnlp-main.576.

S. Bhattamishra, A. Patel, P. Blunsom, and V. Kanade. Understanding In-Context Learning in Transformers and LLMs by Learning to Learn Discrete Functions. In *The Twelfth International Conference on Learning Representations*, 13 Oct. 2023. URL https://openreview.net/forum?id=ekeyCgeRfC.

T. L. Booth and R. A. Thompson. Applying probability measures to abstract languages. *IEEE transactions on computers. Institute of Electrical and Electronics Engineers*, C-22(5):442–450, May 1973. ISSN 0018-9340. doi: 10.1109/t-c.1973.223746. URL https://ieeexplore.ieee.org/document/1672339.

A. Butoi, G. Khalighinejad, A. Svete, J. Valvoda, R. Cotterell, and B. DuSell. Training neural networks as recognizers of formal languages, 11 Nov. 2024. URL http://arxiv.org/abs/2411.07107.

A. Clark. Testing distributional properties of context-free grammars. In S. Verwer, M. van Zaanen, and R. Smetsers, editors, *Proceedings of The 13th International Conference on Grammatical Inference*, volume 57 of *Proceedings of Machine Learning Research*, pages 42–53, Delft, The Netherlands, 2017. PMLR. URL https://proceedings.mlr.press/v57/clark16.pdf.

A. Clark. syntheticpcfg: Code for generating synthetic PCFGs for testing grammatical inference algorithms, 2018. URL https://github.com/alexc17/syntheticpcfg.

DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. Technical report, DeepSeek AI, 22 Jan. 2025.

G. Delétang, A. Ruoss, J. Grau-Moya, T. Genewein, L. K. Wenliang, E. Catt, C. Cundy, M. Hutter, S. Legg, J. Veness, and P. A. Ortega. Neural Networks and the Chomsky Hierarchy. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WbxHAzkeQcn.

N. Dziri, X. Lu, M. Sclar, X. L. Li, L. Jiang, B. Y. Lin, S. Welleck, P. West, C. Bhagavatula, R. L. Bras, J. D. Hwang, S. Sanyal, X. Ren, A. Ettinger, Z. Harchaoui, and Y. Choi. Faith and fate: Limits of transformers on compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Fkckkr3ya8.

M. Finlayson, K. Richardson, A. Sabharwal, and P. Clark. What makes instruction learning hard? An investigation and a new challenge in a synthetic environment, 19 Apr. 2022. URL http://arxiv.org/abs/2204.09148.

Gemma Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, L. Rouillard, T. Mesnard, G. Cideron, J.-B. Grill, S. Ramos, E. Yvinec, M. Casbon, E. Pot, I. Penchev, G. Liu, F. Visin, K. Kenealy, L. Beyer, X. Zhai, A. Tsitsulin, R. Busa-Fekete, A. Feng, N. Sachdeva, B. Coleman, Y. Gao, B. Mustafa, I. Barr, E. Parisotto, D. Tian, M. Eyal, C. Cherry, J.-T. Peter, D. Sinopalnikov, S. Bhupatiraju, R. Agarwal, M. Kazemi, D. Malkin, R. Kumar, D. Vilar, I. Brusilovsky, J. Luo, A. Steiner, A. Friesen, A. Sharma, A. Sharma, A. M. Gilady, A. Goedeckemeyer, A. Saade, A. Feng, A. Kolesnikov, A. Bendebury, A. Abdagic, A. Vadi, A. György, A. S. Pinto, A. Das, A. Bapna, A. Miech, A. Yang, A. Paterson, A. Shenoy, A. Chakrabarti, B. Piot, B. Wu, B. Shahriari, B. Petrini, C. Chen, C. L. Lan, C. A. Choquette-Choo, C. J. Carey, C. Brick, D. Deutsch, D. Eisenbud, D. Cattle, D. Cheng, D. Paparas, D. S. Sreepathihalli, D. Reid, D. Tran, D. Zelle, E. Noland, E. Huizenga, E. Kharitonov, F. Liu, G. Amirkhanyan, G. Cameron, H. Hashemi, H. Klimczak-Plucińska, H. Singh, H. Mehta, H. T. Lehri, H. Hazimeh, I. Ballantyne, I. Szpektor, I. Nardini, J. Pouget-Abadie, J. Chan, J. Stanton, J. Wieting, J. Lai, J. Orbay, J. Fernandez, J. Newlan, J.-Y. Ji, J. Singh, K. Black, K. Yu, K. Hui, K. Vodrahalli, K. Greff, L. Qiu, M. Valentine, M. Coelho, M. Ritter, M. Hoffman, M. Watson, M. Chaturvedi, M. Moynihan, M. Ma, N. Babar, N. Noy, N. Byrd, N. Roy, N. Momchev, N. Chauhan, N. Sachdeva, O. Bunyan, P. Botarda, P. Caron, P. K. Rubenstein, P. Culliton, P. Schmid, P. G. Sessa, P. Xu, P. Stanczyk, P. Tafti, R. Shivanna, R. Wu, R. Pan, R. Rokni, R. Willoughby, R. Vallu, R. Mullins, S. Jerome, S. Smoot, S. Girgin, S. Iqbal, S. Reddy, S. Sheth, S. Põder, S. Bhatnagar, S. R. Panyam, S. Eiger, S. Zhang, T. Liu, T. Yacovone, T. Liechty, U. Kalra, U. Evci, V. Misra, V. Roseberry, V. Feinberg, V. Kolesnikov, W. Han, W. Kwon, X. Chen, Y. Chow, Y. Zhu, Z. Wei, Z. Egyed, V. Cotruta, M. Giang, P. Kirk, A. Rao, K. Black, N. Babar, J. Lo, E. Moreira, L. G. Martins, O. Sanseviero, L. Gonzalez, Z. Gleicher, T. Warkentin, V. Mirrokni, E. Senter, E. Collins, J. Barral, Z. Ghahramani, R. Hadsell, Y. Matias, D. Sculley, S. Petrov, N. Fiedel, N. Shazeer, O. Vinyals, J. Dean, D. Hassabis, K. Kavukcuoglu, C. Farabet, E. Buchatskaya, J.-B. Alayrac, R. Anil, Dmitry, Lepikhin, S. Borgeaud, O. Bachem, A. Joulin, A. Andreev, C. Hardin, R. Dadashi, and L. Hussenot. Gemma 3 Technical Report. Technical report, Google DeepMind, 25 Mar. 2025.

R. Greenlaw, W. L. Ruzzo, and J. Hoover. A compendium of problems complete for P (preliminary). Technical report, 1991. URL https://era.library.ualberta.ca/items/403292c5-460b-49e6-8b05-9a5a7b45b0d6.

K. Gupta, K. Sanders, and A. Solar-Lezama. Randomly sampled language reasoning problems reveal limits of LLMs, 6 Jan. 2025. URL http://arxiv.org/abs/2501.02825.

W. Hua, F. Sun, L. Pan, A. Jardine, and W. Y. Wang. InductionBench: LLMs Fail in the Simplest Complexity Class. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=brw11PScQM.

N. Kim and T. Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.731. URL https://aclanthology.org/2020.emnlp-main.731/.

L. Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *Journal of the ACM*, 49(1):1–15, 1 Jan. 2002. ISSN 0004-5411,1557-735X. doi: 10.1145/505241.505242. URL https://dl.acm.org/doi/10.1145/505241.505242.

W. Merrill and A. Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11: 531–545, 12 June 2023. ISSN 2307-387X. doi: 10.1162/tacl_a_00562. URL https://dx.doi.org/10.1162/tacl_a_00562.

W. Merrill and A. Sabharwal. The Expressive Power of Transformers with Chain of Thought. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, 15 Apr. 2025a. Accessed: 2025-5-12.

OpenAI. OpenAI o3 and o4-mini System Card. https://openai.com/index/o3-o4-mini-system-card/, 16 May 2025b. Accessed: 2025-5-12.

O. Press, M. Zhang, S. Min, L. Schmidt, N. Smith, and M. Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Stroudsburg, PA, USA, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

L. Qiu, L. Jiang, X. Lu, M. Sclar, V. Pyatkin, C. Bhagavatula, B. Wang, Y. Kim, Y. Choi, N. Dziri, and X. Ren. Phenomenal Yet Puzzling: Testing Inductive Reasoning Capabilities of Language Models with Hypothesis Refinement. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=bNt7oajl2a.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research: JMLR*, 21(2020):1–67, 23 Oct. 2019. ISSN 1532-4435,1533-7928. URL http://arxiv.org/abs/1910.10683.

W. L. Ruzzo. Tree-size bounded alternation. *Journal of computer and system sciences*, 21(2): 218–235, 1 Oct. 1980. ISSN 0022-0000,1090-2724. doi: 10.1016/0022-0000(80)90036-7. URL http://dx.doi.org/10.1016/0022-0000(80)90036-7.

V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. V. Nayak, D. Datta, J. Chang, M. T. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Févry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

H. Venkateswaran. Properties that characterize LOGCFL. *Journal of computer and system sciences*, 43(2):380–404, 1 Oct. 1991. ISSN 0022-0000,1090-2724. doi: 10.1016/0022-0000(91)90020-6. URL http://dx.doi.org/10.1016/0022-0000(91)90020-6.

J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Stroudsburg, PA, USA, Oct. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL http://dx.doi.org/10.18653/v1/2020.emnlp-demos.6.

L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, U. Alon, and G. Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

A Related Work

LLMs are typically fine-tuned to follow instructions provided in-context, and can generalize this skill to new instructions that were not included in fine-tuning (Sanh et al., 2022; Wei et al., 2022). It is often difficult, however, to automatically verify whether the model indeed followed the instructions. Like our work, the IFEval benchmark (Zhou et al., 2023) addresses the verification issue by placing formal constraints on the output (e.g., avoiding commas), but it does not evaluate complex and long instructions as we do. Our focus on evaluating compositionality is also related to work on compositionality in question answering (Dziri et al., 2023; Press et al., 2023) and semantic parsing (Kim and Linzen, 2020).

A number of studies have evaluated LLMs' ability to recognize formal languages (Finlayson et al., 2022; Bhattamishra et al., 2020; Delétang et al., 2023; Butoi et al., 2024). For example, Finlayson et al. (2022) evaluated the ability of T5 (Raffel et al., 2019) to recognize regular expressions, and found that the model's performance degraded as languages became more complex. Our study differs from theirs both in that we study a richer class of languages, where recognition requires more compositional computation, and in that we evaluate zero-shot instruction following, without fine-tuning the model to recognize a particular grammar. In this sense, our work is more similar to that of Gupta et al. (2025), who evaluated LLMs on an in-context version of transduction and completion tasks for regular languages, finding that some regular languages can pose challenges for currently-available LLMs. Our work is also distinct from studies where models are expected to induce the grammar of a language from positive examples (Akyürek et al., 2024; Hua et al., 2025; Qiu et al., 2024; Bhattamishra et al., 2023).

B Example Task Prompt

User prompt for RELIC

```
You will be presented with a context-free grammar in Chomsky normal form and a
string which may or may not be in the language defined by the given grammar.  Your
job is to determine whether or not the grammar generates the provided string.  You
can use any reasoning strategy you like, but you must end your response with either
'Yes' (if the string is generated by the grammar) or 'No' (if it isn't.)

Grammar:
‘‘‘
S -> NT97 NT1
NT180 -> NT47 NT121
NT120 -> NT73 NT121
NT114 -> NT197 NT79
NT191 -> NT76 NT49
NT8 -> NT90 NT28
NT192 -> NT140 NT152
...
...
NT171 -> 't59'
NT31 -> 't139'
NT172 -> 't28'
NT100 -> 't16'
NT187 -> 't158'
NT100 -> 't44'
...
...
‘‘‘

Here is the string you need to evaluate:

String:  ‘t64‘.

Remember, end your response with either 'Yes' or 'No'.
```

C Grammar and Example Information

C.1 Distributional Statistics for Grammars and Examples

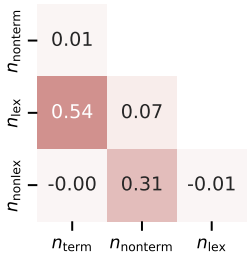

Figure 4: Correlations between generating hyperparameters for the released static set. Note that n_{lex} and n_{term} are inherently correlated.

To construct the static evaluation set used in our experiments, we first over-generate grammars ($\sim$1000 total) where the four generating hyperparameters of each grammar (the numbers of terminal n_{term} and nonterminal n_{nonterm} symbols and the numbers of lexical n_{lex} and nonlexical n_{nonlex} production rules) are bounded above by 500. We then subsample these grammars to include the 200 grammars whose hyperparameters are minimally correlated with one another. Figure 4 below reports the hyperparameter correlations in the released set. Note that the correlations between $n_{\text{term}} \sim n_{\text{lex}}$ and $n_{\text{nonterm}} \sim n_{\text{nonlex}}$ are higher than the others since the former term is bounded above by the latter in both cases.

From each grammar we sample positive and negative strings. Positive strings are sampled by converting each grammar into a probabilistic context-free grammar (Booth and Thompson, 1973) with uniform probability for each right-hand-side path among all productions which share a left-hand-side non-terminal and sampling a production stochastically. We over-sample positive strings and filter so that they are of length at most 50, and such that we have no more than 10 strings per length. Negative strings are sampled by drawing strings over the set of terminal symbols Σ^+ of fixed length $1 \leq \ell \leq 50$ uniformly-at-random and rejecting any strings which are parseable by the grammar. We repeat this process until we have 10 strings per length.

Since positive examples are not drawn with a pre-determined length and not all grammars can generate 10 strings for each length, the resulting set of sampled strings will in some cases be smaller than that of the negative examples; fig. 5 shows the relative proportions of positive and negative samples drawn from the released set of grammars. For our evaluations, we choose not to post-hoc rebalance example types for each length since the distribution of positive examples by length is a property of the grammar. Since not all grammars will generate strings of every length in equal proportions, the length of an example contains relevant information about the example's type, albeit information which is not provided to the model independently from the grammar itself. For a model to justifiably use the example length to arrive at the correct answer, it must derive the relevant properties from the production rules itself.

We refer to the size of a grammar's examples, relative to the theoretical maximum of 1000 strings (= 2 example types * 50 lengths * 10 examples per length per type) as the grammar's *coverage*; fig. 6 shows the coverage of the grammars used in our experiments.

D Detailed Results

E Inference Cost, Setup, and Hyperparameters

Evaluations on OpenAI models used roughly $15k total of compute credits. Evaluations on the open-weights models were run on a GPU cluster using Nvidia A100s, H100s V100s, and RTX8000s; models were loaded using the HuggingFace `transformers` library (Wolf et al., 2020).

Table 3 reports the inference hyperparameters used in our experiments. For the open-weights models, which we run on local hardware, we restrict the number of new tokens (`max_completion_tokens`) to 4096 to limit memory usage and inference time. For all models, we generate completions with sampling using the default parameters (temperature τ and nucleus constant p) specified by the model or API.

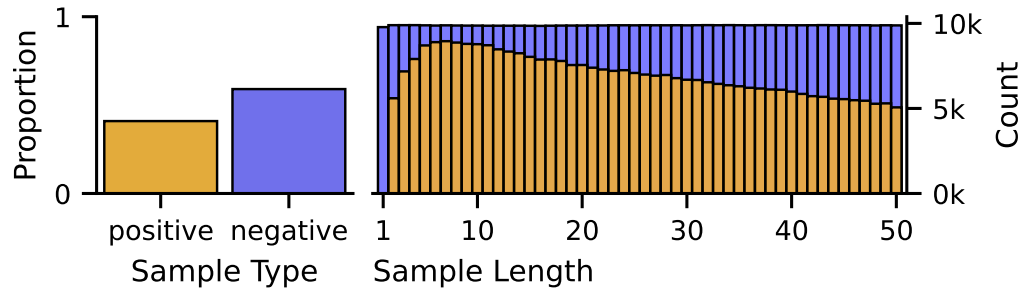

Figure 5: Proportions of example types represented in the static dataset, in aggregate (**left**) and broken down by example length (**right**).

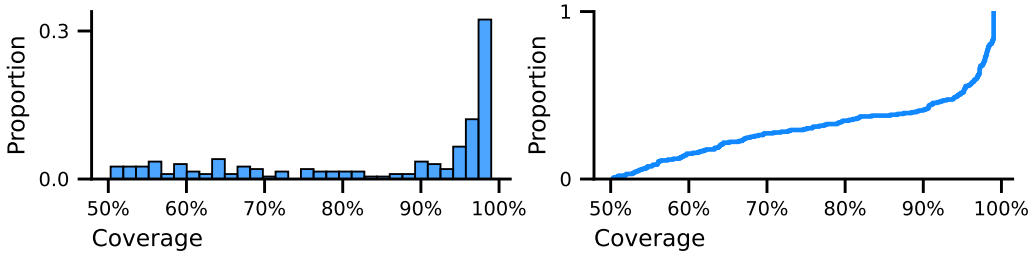

Figure 6: Distribution of grammars by coverage (i.e., the size of the language they generate, measured as the number of positive examples of lengths $\ell \leq 50$, with a maximum of 10 examples/length, out of a theoretical maximum of 500) shown as a histogram (**top**) and a cumulative distribution function (**bottom**).

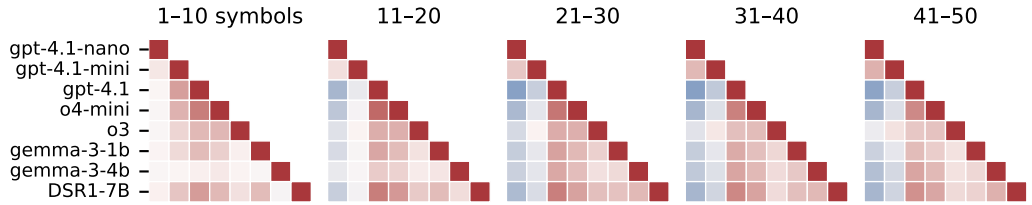

Figure 7: Spearman's rank correlation coefficients for the per-example accuracies between different models, faceted by example length. On short examples of length ≤ 10, all models are moderately correlated with one another; on longer examples, `gpt-4.1-nano` and `gpt-4.1.mini` become less correlated with the other models.

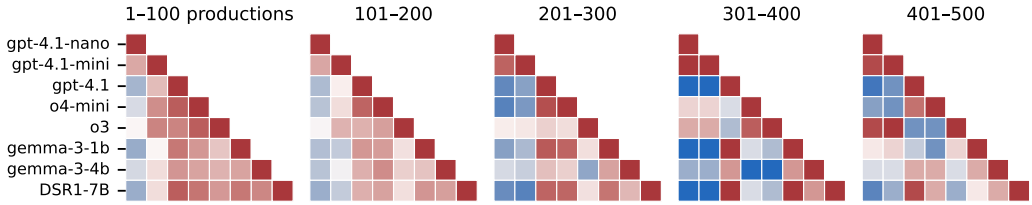

Figure 8: Spearman's rank correlation coefficients for the per-example accuracies between different models, faceted by example length. On short examples of length ≤ 10, all models are moderately correlated with one another; on longer examples, `gpt-4.1-nano` and `gpt-4.1.mini` become less correlated with the other models.

		$\log(n_{\text{term}})$	$\log(n_{\text{lex}})$	$\log(n_{\text{nonterm}})$	$\log(n_{\text{nonlex}})$
`gpt-4.1-nano`	r_{F1}	0.26	0.23	0.14	−0.10
	$r_{\text{Acc.}}$	0.15	0.17	0.09	−0.05
`gpt-4.1-mini`	r_{F1}	0.07	−0.02	−0.13	−0.57
	$r_{\text{Acc.}}$	0.09	0.02	−0.13	−0.45
`gpt-4.1`	r_{F1}	−0.11	−0.37	−0.52	−0.59
	$r_{\text{Acc.}}$	−0.13	−0.28	−0.25	−0.27
`o4-mini`	r_{F1}	−0.20	−0.39	−0.60	−0.79
	$r_{\text{Acc.}}$	−0.15	−0.28	−0.20	−0.36
`o3`	r_{F1}	−0.12	−0.22	−0.37	−0.75
	$r_{\text{Acc.}}$	−0.06	−0.12	−0.17	−0.40
`gemma-3-1b`	r_{F1}	−0.21	−0.24	−0.19	−0.20
	$r_{\text{Acc.}}$	−0.09	−0.13	−0.13	−0.11
`gemma-3-4b`	r_{F1}	−0.22	−0.36	−0.49	−0.61
	$r_{\text{Acc.}}$	−0.05	−0.09	−0.21	−0.19
`DSR1-7B`	r_{F1}	−0.14	−0.24	−0.43	−0.38
	$r_{\text{Acc.}}$	−0.12	−0.20	−0.20	−0.20

Table 1: Pearson correlation coefficients r between models' accuracy/macro F1 scores and grammar hyperparameters, including the compression ratio CR of positive examples and the numbers of terminal symbols n_{term}, non-terminal symbols n_{nonterm}, lexical productions n_{lex}, and non-lexical productions n_{nonlex}. Correlation scores are taken over the mean F1 and accuracy values grouped by grammar.

Term	**Coeff.**	p**-value**	**Sig.**
Intercept	46.56	< 0.001	***
`gpt-4.1-mini`	13.18	< 0.001	***
`gpt-4.1`	19.43	< 0.001	***
`o4-mini`	21.69	< 0.001	***
`o3`	26.26	< 0.001	***
`gemma-3-1b`	7.87	< 0.001	***
`gemma-3-4b`	7.38	< 0.001	***
`DSR1-7B`	9.66	< 0.001	***
$\log_{10}(n_{\text{nonlex}})$*`gpt-4.1-nano`	−2.28	< 0.001	***
$\log_{10}(n_{\text{nonlex}})$*`gpt-4.1-mini`	−24.17	< 0.001	***
$\log_{10}(n_{\text{nonlex}})$*`gpt-4.1`	−10.28	< 0.001	***
$\log_{10}(n_{\text{nonlex}})$*`o4-mini`	−14.41	< 0.001	***
$\log_{10}(n_{\text{nonlex}})$*`o3`	−23.54	< 0.001	***
$\log_{10}(n_{\text{nonlex}})$*`gemma-3-1b`	−4.44	< 0.001	***
$\log_{10}(n_{\text{nonlex}})$*`gemma-3-4b`	−9.93	< 0.001	***
$\log_{10}(n_{\text{nonlex}})$*`DSR1-7B`	−9.78	< 0.001	***
$\log_{10}(\ell)$*`gpt-4.1-nano`	−26.79	< 0.001	***
$\log_{10}(\ell)$*`gpt-4.1-mini`	0.72	0.385	
$\log_{10}(\ell)$*`gpt-4.1`	15.93	< 0.001	***
$\log_{10}(\ell)$*`o4-mini`	8.73	< 0.001	***
$\log_{10}(\ell)$*`o3`	−1.36	0.102	
$\log_{10}(\ell)$*`gemma-3-1b`	28.37	< 0.001	
$\log_{10}(\ell)$*`gemma-3-4b`	41.20	< 0.001	***
$\log_{10}(\ell)$*`DSR1-7B`	28.03	< 0.001	***
$\log_{10}(n_{\text{nonlex}})*\log_{10}(\ell)$	−10.28	< 0.001	***

Table 2: Regression coefficients for accuracy by model, $\log_{10}(n_{\text{nonlex}})$, and example length ℓ. The base level of the model factor is `gpt-4.1-nano`. Significance values are listed as *** for $p < 0.001$, ** for $p < 0.01$, and * for $p < 0.05$.

Model	New tokens	τ	p
`gpt-4.1-nano`	None	1.0	1.00
`gpt-4.1-mini`	None	1.0	1.00
`gpt-4.1`	None	1.0	1.00
`o4-mini`	None	1.0	1.00
`o3`	None	1.0	1.00
`gemma-3-1b-it`	4096	1.0	0.95
`gemma-3-4b-it`	4096	1.0	0.95
`DeepSeek-R1-Distill-Qwen-7B`	2048	0.6	0.95

Table 3: Inference hyperparameters used in experiments.

F Chains-of-Thought

This appendix provides a few examples of the chains-of-thought (token generated between the prompt and the answer) produced by two of the models, and `DeepSeek-R1-Distill-Qwen-7B` (which we abbreviate here as DSR1). We elide some text with `...` and some paragraphs with `[...]` for brevity.

F.1 Example A

gpt-4.1-mini
Grammar 20250415162507_949738

```
Let's analyze the grammar carefully and the string to verify if the string can be
generated by the grammar.
                                        [...]
### Step 1:  Since S -> NT5 NT2
The string breaks into two substrings:
- substring generated by NT5 - substring generated by NT2
Let's try possible splits:
- split after 1 token:  (t30) (t24 t24 t23 t4)
- split after 2 tokens:  (t30 t24) (t24 t23 t4)
- split after 3 tokens:  (t30 t24 t24) (t23 t4)
- split after 4 tokens:  (t30 t24 t24 t23) (t4)
Try each possibility.

### Step 2:  Can NT5 generate the initial substring?
**NT5 options:**
- NT5 -> NT0 NT5
- NT5 -> terminals like 't18', 't23', 't25', 't13' (no 't30' here)
**NT0 terminals:**
                                        [...]
- 't24' (yes)
- 't30' (yes) So 't30' is in NT0 terminals.

Try first split:
                                        [...]
NT5 is not nullable (no empty production), so this is not possible.

Try split after 2 tokens:
                                        [...]
So no.

Try split after 3 tokens:
                                        [...]
No.

Try split after 4 tokens:
```

```
NT5 = 't30 t24 t24 t23'

Try parsing tokens stepwise:

- t30 as NT0 (allowed)
remaining t24 t24 t23 as NT5

Try NT5->NT0 NT5 again:

- t24 as NT0 (allowed)
remaining t24 t23 as NT5

Again:

- t24 as NT0 (allowed)
remaining t23 as NT5

Check 't23' in NT5 terminals?  Yes!  NT5 -> 't23' allowed.

So:

NT5 derivation:
t30 (NT0)
t24 (NT0)
t24 (NT0)
t23 (NT5 terminal)

Wait, but this would only work if NT5 allows chaining of:

NT5 -> NT0 NT5 (>=1)

Ending with NT5 terminal.

So parse:

- NT5:
NT5 -> NT0 NT5
NT0 -> 't30'
NT5 -> NT0 NT5
NT0 -> 't24'
NT5 -> NT0 NT5
NT0 -> 't24'
NT5 -> 't23' (terminal)

Tokens:  t30 t24 t24 t23

So yes, this derives the 4 tokens.

### Step 3:  Now check if NT2 can generate last token:  't4'

NT2 terminals:  't9', 't23', 't27', 't4', 't30'

Yes, 't4' is a terminal allowed by NT2
Therefore, NT2 can directly generate 't4'

### Step 4:  Full derivation

S -> NT5 NT2

NT5 -> 't30 t24 t24 t23' (via recursive NT5 -> NT0 NT5)
NT2 -> 't4' terminal
```

```
The full string ‘t30 t24 t24 t23 t4‘ is generated by S.

### Final conclusion:

The string can be generated by the grammar.

**Answer:**

Yes
```

F.2 Example A

DSR1 Grammar 20250218222557

```
...  I think the approach is to start from the string and see if I can break it down
into the non-terminals according to the production rules.  ...                    ¶1
                                        [...]
...  since the string has 41 symbols, which is an odd number.  Wait, let me count the
tokens:  the string is "t75 t104 t120 t27 t137 t27 t11 t11 t11 t173 t181 t27 t84 t11
t82 t104 t120 t168 t69 t141 t158 t9 t12 t140 t102 t114 t190 t141 t147 t79 t141 t159
t178 t183 t183 t83 t141 t158 t9 t55 t74 t170 t140 t102 t114 t190".  Let me count these:
41 tokens.  ...                                                                  ¶2
                                        [...]
Alternatively, starting from the beginning, the first symbol is ’t75’, which must
come from some terminal in the grammar.  Looking at the grammar, let’s find which
non-terminals have a production ending with ’t75’.  Wait, in the grammar, NT44 ->
’t132’, and NT197 -> ’t178’, and NT97 -> ’t75’ according to the end of the grammar
list:  NT97 -> ’t75’.  So, NT97 can directly produce ’t75’.                      ¶3
                                        [...]
Now, looking at NT1:  according to the grammar, NT1 -> ’t1’.  So, that’s a terminal.
So the first two symbols in the string must be ’t75’ followed by ’t1’.  Let me check
the given string:  the first two tokens are ’t75 t104’, but wait, that’s not ’t75’
followed by ’t1’.  So that’s a problem.                                          ¶4
                                        [...]
Wait, alternatively, maybe the NT1 is not the immediate right child but another
expansion.  ...                                                                  ¶5
                                        [...]
Therefore, it’s not possible that the entire string is generated by this grammar
because the second symbol doesn’t match ’t1’.  ...                               ¶6
```

F.3 Example B

DSR1 Grammar 20250402155408_676876

```
                                        [...]
Looking at the patterns, I see that ‘t136‘ might correspond to multiple parts.  I
remember that ‘t136‘ could link to ‘t39‘ and ‘t6‘ based on some earlier patterns.  ¶1
                                        [...]
Putting it all together, it seems like ‘t136‘ is a valid node in the context.  So, the
answer should be ’Yes’.                                                          ¶2
```

F.4 Example C

DSR1 Grammar 20250402155408_676876

```
[...]
First, I see that the task involves checking if the token 't27' appears in the output
of a certain string evaluation.  The string to evaluate is `t44 t249 t80 t333 t171
t194 t329 t224 t196 t64 t8 t43 t16 t15 t280 t220 t107 t230 t334 t207 t19 t296`.   ¶1
[...]
So, based on my evaluation, the output of the evaluation doesn't contain 't27'.
Therefore, the condition isn't met, and the answer should be 'No'.                ¶2
```

G Analysis of Strategies using the LLM-as-a-Judge Paradigm

To categorize the strategies used by the `gpt-4.1-` series of models, we instruct `o4-mini` to classify model completions into one of three categories (following the 'LLM-as-a-judge' framework of Zheng et al. 2023): *rule-based*, where the model provides a complete derivation for the example or proves that one cannot exist; *heuristic*, where the model appeals to distributional properties of the example or partial derivations to argue that a string is (un)likely to be generated by the grammar; or *code*, where the model defines a program or algorithm to solve the task and then hallucinates 'running' the function to arrive at the result. We use the following prompt:

User prompt for strategy classification

```
You will be presented with a completion from an LLM which was given a context-free
grammar and a string of symbols drawn from that grammar's set of terminal symbols and
asked to determine whether the string is generated by the grammar or not.  Your job is
to classify how the LLM attempted to solve the task by binning the completion strategy
into one of the following categories:
- `heuristic`:  The LLM attempts to solve the task by using heuristics it surmises
from the grammar, such as ``if the string is long, it is likely generated by the
grammar'' or ``the string only contains terminal symbols present in the grammar, so
it's likely a positive sample''.  Count strategies as heuristic if they appeal to the
existence of certain production rules but do not rigorously determine that no such
derivation exists.
- `rule-based`:  The LLM attempts to solve the task by writing out the FULL DERIVATION
of the sample from the grammar, or rigorously determining that no such derivation
exists.  Only count strategies as rule-based if the LLM doesn't use any guesswork to
reach its final conclusion.
- `code`:  The LLM attempts to solve the task by writing out a program or algorithm
which it claims will solve the task.  This includes writing out a program in a
programming language, or writing out pseudocode.

You can write as much as you want in your answer, but please end your response with the
name of the classification you think is most appropriate.

Here is the LLM's completion:

```
{completion}
```
```

To validate the accuracy of `o4-mini`'s categorizations of model responses, we first prompt `o4-mini` to categorize the completions for a single grammar. We then sample 20 (*model response*, *predicted category*) pairs and have three of the paper authors each label the model responses according to the same rubric. We then compute the inter-annotator agreement, majority-annotator categories, and `o4-mini`'s agreement with the majority class. We attain an inter-annotator agreement of 93.3%, and `o4-mini` attains a majority-class agreement of 70% and a weighted-class agreement of 73.3%, as shown in table 4. We qualitatively note that `o4-mini` is stricter than the three annotators at

categorizing responses as *rule-based*; all instances of disagreement with the majority class are cases when annotators labeled a response as *rule-based* while `o4-mini` labeled it as *heuristic*.

	A	**B**	**C**	**Majority**	`o4-mini`	**IAA**	**MA**	**WA**
1	heuristic	rule-based	rule-based	rule-based	heuristic	0.667	0	0.333
2	rule-based	rule-based	rule-based	rule-based	rule-based	1	1	1
3	rule-based	rule-based	rule-based	rule-based	rule-based	1	1	1
4	rule-based	rule-based	rule-based	rule-based	heuristic	1	0	0
5	heuristic	rule-based	rule-based	rule-based	heuristic	0.667	0	0.333
6	heuristic	heuristic	heuristic	heuristic	heuristic	1	1	1
7	rule-based	rule-based	rule-based	rule-based	rule-based	1	1	1
8	heuristic	rule-based	rule-based	rule-based	heuristic	0.667	0	0.333
9	rule-based	rule-based	rule-based	rule-based	rule-based	1	1	1
10	rule-based	rule-based	rule-based	rule-based	rule-based	1	1	1
11	heuristic	rule-based	rule-based	rule-based	rule-based	0.667	1	0.667
12	heuristic	heuristic	heuristic	heuristic	heuristic	1	1	1
13	rule-based	rule-based	rule-based	rule-based	heuristic	1	0	0
14	rule-based	rule-based	rule-based	rule-based	rule-based	1	1	1
15	rule-based	rule-based	rule-based	rule-based	heuristic	1	0	0
16	rule-based	rule-based	rule-based	rule-based	rule-based	1	1	1
17	heuristic	heuristic	heuristic	heuristic	heuristic	1	1	1
18	rule-based	rule-based	rule-based	rule-based	rule-based	1	1	1
19	rule-based	rule-based	rule-based	rule-based	rule-based	1	1	1
20	heuristic	heuristic	heuristic	heuristic	heuristic	1	1	1
						0.933	0.7	0.733

Table 4: Validation of `o4-mini`'s categorizations for 20 model responses. We report inter-annotator-agreement (IAA) among the three annotators, the majority agreement (MA) between `o4-mini` and the annotator majority, and the weighted agreement (WA) between `o4-mini` and the three annotators.