RELIC: Evaluating Complex Reasoning via the Recognition of Languages In-Context

Jackson Petty[λ*] **Michael Y. Hu**[δ] **Wentao Wang**[δ]
Shauli Ravfogel[δ] **William Merrill**[δ] **Tal Linzen**[λδ]

Department of Linguistics[λ] Center for Data Science[δ]
New York University

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

Abstract

Large language models (LLMs) are increasingly used to solve complex tasks where they must retrieve and compose many pieces of in-context information in long reasoning chains. For many real-world tasks it is hard to accurately gauge how model performance and strategy change as task complexity grows. To evaluate models' complex reasoning capability in a scalable and verifiable way, we introduce RELIC (Recognition of Languages In-Context), a framework that evaluates an LLM's ability to decide whether a given string belongs to the context-free language (CFL) generated by a grammar presented in-context. CFL recognition allows us to modulate the intrinsic complexity of the problem by varying grammar size and string length and translate this asymptotic complexity into predictions for ideal LLM performance. We find that even the most advanced reasoning models perform poorly on RELIC, not only failing to appropriately scale their inference compute to keep pace with task difficulty, but even reducing the number of reasoning tokens they use as task complexity increases. We find that these decreases in compute accompany changes in reasoning strategy, as models move from identifying and implementing algorithmic solutions to guessing. For models whose full completions go uninspected, this manifests as "quiet quitting" on hard tasks.

Code: https://jpetty.org/relic

1 Introduction

Large language models (LLMs) are increasingly used to solve problems "zero-shot," using only a task specification provided in the context window, without labeled examples or additional training. As these problems grow in complexity, models must reason through increasingly many interdependent steps and reference more and more pieces of in-context information; for instance, even a simple question like "using my credit card statement, tell 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 many different items from its context and perform multiple operations in a particular sequence (in this example, averaging them and comparing the outputs).

Formal domains, where solutions can be verified by an external system, are a valuable lens into LLMs' performance on complex reasoning tasks (Lambert et al., 2025). By studying reasoning problems in these domains, we can leverage our analytic understanding of the complexity of the formal problem to derive predictions for an optimal reasoner's performance on each instance of the task, and compare these predictions to the LLMs' empirical performance in a precise quantitative way.

Here, we propose the **RE**cognition of **L**anguages **I**n-**C**ontext (**RELIC**) framework, which applies this approach to *language recognition*, a well-studied paradigm from theoretical computer science and computational linguistics. This paradigm consists of determining whether a given (formal) grammar generates a given string. In RELIC, an LLM is prompted with an arbitrary grammar and a string drawn from that grammar's terminal symbols and is asked to accept or reject the string based on the rules of the grammar.

We focus on language recognition for two complementary reasons. First, this problem is well-studied in a classical algorithmic setting, allowing us to predict how varying the hyperparameters which drive intrinsic task complexity, such as grammar size and string length, may affect performance.

Second, language recognition, though formally defined, has connections (§2.3) to fuzzier 'real-world' reasoning capabilities which are harder to study in their own right and which do not come equipped with precise controllability, such as *in-*

* Correspondence to research@jacksonpetty.org.

Transactions of the Association for Computational Linguistics, vol. 14, pp. 1431–1452, 2026. https://doi.org/10.1162/TACL.a.735
Action Editor: Ehud Reiter. Submission batch: 2/2025; Revision batch: 8/2025; Published 7/2026.
© 2026 Association for Computational Linguistics. Distributed under a CC-BY 4.0 license.

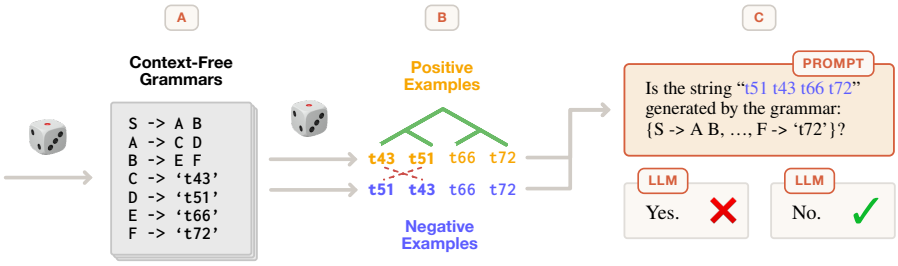

Figure 1: RELIC is an evaluation framework for complex reasoning, where we **(a)** stochastically generate context-free grammars of given complexities, **(b)** sample positive and negative strings for each grammar, and **(c)** prompt models to classify whether the strings are generated by those grammars.

struction following (Wei et al., 2022a; Finlayson et al., 2022) and *in-context language translation* (Tanzer et al., 2023). Like these problems, RELIC requires sophisticated use of context: multi-hop retrieval of self-referential pieces (retrieving additional pieces based on ones already retrieved); composition of these pieces into candidate solutions; and a search over the candidate space for valid solutions. It thus serves as a formal analogue to these fuzzier tasks while bringing the benefit of easy verification.

In this work we instantiate the RELIC paradigm to explore for the first time how well LLMs can recognize *context-free languages* (CFLs) in-context. CFLs are a class of formal languages designed to mimic the compositional structure of both programming and natural languages (Chomsky and Schützenberger, 1963), the two primary interaction media for LLMs. The asymptotic complexity of CFL recognition (§2.1) is within the capacity of transformers with chains of thought that can vary in length (Greenlaw et al., 1991; Merrill and Sabharwal, 2023); in particular, to solve the problem in all cases, the length of the chain of thought must be allowed to grow linearly in the size of the grammar and superlinearly in the length of the string. But this is only a theoretical possibility: it is an empirical question whether currently available LLMs, most of which are able to grow their chains of thought in this way, are able to solve this problem in practice.

We create a framework for stochastically generating novel context-free grammars (CFGs) and drawing positive and negative samples from these grammars. Using this framework, we construct a benchmark of grammars of varying size (up to 500 nonterminal productions) and accompanying sets of positive and negative example strings (of lengths of up to 50 symbols). We evaluate a range of LLMs on this benchmark, including smaller open-weight models (Gemma 3 and a distilled version of DeepSeek-R1) and proprietary models from OpenAI (`o3`, `o4-mini` and the `gpt-4.1` family).

To preview our findings, most models perform well on small grammars and short strings, indicating a general understanding of the task; but all models, including the most advanced reasoning models, fall to 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. Qualitative analysis of the chain-of-thought (CoT) traces shows that in many cases models increasingly rely on incorrect heuristics; quantitatively, the lengths of CoTs do not grow as needed for the correct strategy to perform the task. For "closed reasoning" models that don't return full reasoning traces—like `o4-mini` and `o3`—this manifests as "quiet quitting" where models silently shift to prefer invalid reasoning strategies as problems get harder.

These results indicate that LLM reasoning is brittle, even among state-of-the-art reasoning models. We conclude that there is substantial headroom for improvement on RELIC and, by extension, on the fuzzy problems for which language recognition is a formal analogue. The observation that models "quiet quit" motivates finding ways to ensure that LLMs stay faithful to valid reasoning strategies while mitigating the associated compute costs. More generally, these results illustrate the potential for using formal languages to produce evaluations that are resistant to data contamination, and whose difficulty can be precisely scaled as LLMs' capabilities improve.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

2 Background

2.1 LLMs and Complexity Analysis

We are inspired by a line of work that considers how intrinsic complexity measures for formal tasks determine how efficiently a model can solve the tasks based on the model's architecture, hyperparameters, and the number of test-time compute tokens. Prior work has examined transformers through the lens of circuit complexity, establishing that transformers without CoT (i.e., without the ability to produce intermediate tokens before responding) are in the complexity class TC^0 (Merrill and Sabharwal, 2023), while those with linearly-many CoT steps are in the complexity class NC^1 (Merrill and Sabharwal, 2024).

The recognition of membership in a context-free grammar without ε-productions is known to be in the complexity class AC^1 and to be NC^1-hard[1] (Ruzzo, 1980; Venkateswaran, 1991; Greenlaw et al., 1991). Standard context-free parsing algorithms run in time $\Theta(G \cdot \ell^3)$ with little likelihood of significant improvement (where G is the number of rules in the grammar and ℓ is the length of the string; Lee 2002). Translating this runtime-complexity into steps of inference (Merrill and Sabharwal, 2024), this means that transformers with CoT can, in theory, solve RELIC, but at a steep cost: the number of CoT steps required is bounded between at least $\Omega(G \cdot \ell^{1.7})$ and at most $O(G \cdot \ell^6)$. Since inference compute is, in practice, a limited quantity, we then predict that there are complexity bounds on both grammars and strings beyond which transformer LLMs will be unable to perform above chance.

2.2 LLMs and Context Use

In addition to the ability to execute algorithms in line with theoretical measures of their difficulty, to solve complex tasks provided in a prompt, LLMs require the more prosaic ability to effectively use information provided in-context.

Early work on LLMs' context-use ability focused on retrieval tasks like "needle-in-the-haystack" evaluations where models are tasked with retrieving specific pieces of information provided in-context (Kamradt, 2023; Hsieh et al., 2024; Bohnet et al., 2024; Zhang et al., 2024; Arora et al., 2024). More difficult tests for context use introduce elements of *referentiality* and *compositionality*[2] (piecing together multiple pieces of retrieved information). Kim and Linzen (2020) and Press et al. (2023), *inter alia*, demonstrate the challenge that compositional reasoning poses for language models in both toy and real-world settings.

More recent work in this direction includes Wu et al. (2025), which provides an account of how language models learn to implement variable-binding, a necessary component of compositional reasoning; NeedleBench (Li et al., 2025); and Michelangelo (Vodrahalli et al., 2024). Like RELIC, Michelangelo consists of synthetically generated tasks, which make it possible to control the complexity of the task and address dataset leakage issues. Our work is motivated by similar concerns but studies the substantially more challenging task of language recognition—a setting which is harder both in the formal complexity-theory sense addressed above, and because unlike these datasets RELIC is characterized by *non-monotonicity* (the order in which pieces of the context must be retrieved or modified may not be the same as their appearance order) and *search* (multiple candidate solutions must be constructed from retrieved pieces and evaluated to explore the solution space; Yao et al. 2023; Yang et al. 2023).

2.3 LLMs and Reasoning

Poor performance of pure language models on tasks requiring reasoning motivated the development of increasingly sophisticated approaches to tackle complex problems, notably by scaling the amount of compute models expend at inference time (test-time compute), first through methods like chain-of-thought prompting (Wei et al., 2022b), and later through iterated self-reflection (OpenAI 2024) in reasoning language models like OpenAI's o1, o3, o4-mini, and GPT-5 (OpenAI et al., 2024; OpenAI, 2025c,a); Gemini Thinking (Google, 2025); Claude Thinking (Anthropic, 2025); and DeepSeek-R1 (DeepSeek-AI et al., 2025). These approaches

[1] TC^0 problems are those decidable by circuits of constant depth and polynomial size, containing binary and majority gates; $\mathsf{NC}^1 \supseteq \mathsf{TC}^0$ problems are decidable by circuits of logarithmic depth and polynomial size, with each gate having at most two inputs; $\mathsf{AC}^1 \supseteq \mathsf{NC}^1$ problems are decidable by circuits of logarithmic depth and polynomial size, but with no fan-in restrictions on gates.

[2] We mean compositionality in a broad sense of combining atomic primitives in some well-defined manner; the more specific sense of deriving the *meaning* of expressions from their constituent parts is implicated in question answering (Dziri et al., 2023; Press et al., 2023) and semantic parsing (Kim and Linzen, 2020).

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

have led to broad improvements in performance across a wide array of static reasoning benchmarks. Yet recent work has also noted the brittleness in reasoning performance (Shojaee et al., 2025), where even the best models fail to implement consistent algorithmic solutions in many cases or to generalize to new instances or kinds of tasks.

Two classes of problem of particular relevance to RELIC are *instruction following* and *in-context translation*. Regarding the former, 'base' LLMs are typically fine-tuned to follow in-context instructions, and can generalize this skill to new instruction sets not seen during training (Sanh et al., 2022; Wei et al., 2022a). Language recognition was first proposed as a formal analogue to instruction following by Finlayson et al. (2022), who note that grammars function as instruction sets for the task of (regular) language recognition. RELIC expands on work in this direction by focusing on a richer class of languages, CFLs, which have a higher level of intrinsic complexity than regular languages do and consequently are better able to model the kinds of compositional structures present in both natural and programming languages (Chomsky, 1959; Chomsky and Schützenberger, 1963). Unlike work on LLM language recognition where the LLMs were trained for the specific task or on specific grammars (Bhattamishra et al., 2020; Delétang et al., 2023; Butoi et al., 2025), we examine the ability of general-purpose LLMs to accept or reject strings.

The second class of problems that is closely related to RELIC, *in-context translation*, was first proposed by Tanzer et al. (2023). In this task, an LLM is prompted with a complete description of a language (e.g., a reference grammar or textbook) and is expected to use this description to translate into a low-resource language that was not included in its pretraining corpus. This experimental design, while innovative, suffers from the difficulty of verifying the correctness of low-resource natural language translations and leaves open the question of how LLM performance is impacted by the language description's complexity. Our work here begins to address some of these concerns. As an instance of language *recognition*, RELIC is closely related to language translation, and is the formal analogue to the task of determining whether or not a translated sentence is *grammatical*, independent from whether or not it is the correct translation. Very recently Gupta et al. (2025) has begun to explore in-context translation performance through formal means, but only for regular languages too simple to be good proxies for natural, human languages.

3 RELIC: Recognizing Languages In-Context

RELIC is a framework for generating synthetic benchmarks: it can be used to generate new evaluation instances of increasing complexity that are extremely unlikely to have been observed by the model in training. In this paper, in addition to implementing this process in a codebase, we also generate a static benchmark using this process, and evaluate contemporary language models on the benchmark.

Following Clark (2017, 2018), we stochastically generate grammars which are parameterized by four values: the number of terminal symbols n'_{term}, nonterminal symbols n'_{nonterm}, lexical production rules n'_{lex}, and nonlexical production rules n'_{nonlex}. We start by defining n'_{term} symbols $\Sigma = \{\mathtt{t}_1, \mathtt{t}_2, \dots\}$; and n'_{nonterm} symbols $V = \{\mathtt{NT}_1, \mathtt{NT}_2, \dots\}$. We then sample n'_{lex} lexical production rules $\mathtt{NT}_a \rightarrow$ '$\mathtt{t}_b$' from the set of pairs $V \times \Sigma$, and we sample n'_{nonlex} nonlexical production rules $\mathtt{NT}_a \rightarrow \mathtt{NT}_b\ \mathtt{NT}_c$ from the set of all triples $(\{\mathtt{S}\} \cup V) \times V \times V$, where S is a privileged start symbol. The grammar is then trimmed to remove any nonlexical rules which do not lead to a lexical rule, and any lexical rules which are inaccessible from nonlexical productions. The result is a reduced context-free grammar $\mathbf{G}$ with at most as many terminals n_{term}, nonterminals n_{nonterm}, lexical productions n_{lex}, and nonlexical productions n_{nonlex} as the generating parameters; for analysis, we measure the grammar's complexity according to these values after reduction, and specifically define its size as $G = n_{\text{lex}} + n_{\text{nonlex}}$.

Using this framework, 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. To reduce the correlation between the four parameters, we oversample many grammars and filter to obtain the 200 with minimally-correlated parameters; we note that some parameter pairs, such as n_{lex} and n_{term}, are inherently correlated and it is not possible to completely decorrelate them (see table 1 for the correlation coefficients).

For each grammar, we aim to sample 10 positive strings $\mathbf{s}$ of lengths $1 \leq \ell \leq 50$ by treating each

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

grammar as a probabilistic context-free grammar with uniform probability assigned to production rules sharing a left-hand symbol. We also generate 10 negative strings (i.e., strings which are not generated by the grammar) of each length: we sample negative strings from a unigram model over the terminal symbols Σ^+ and reject any strings which are parseable by the grammar.[3] Not all randomly-generated grammars will be able to produce arbitrarily many strings of a given sequence length. We refer to the size of the set of generated examples relative to a defined goal of 1000 examples as the grammar's *coverage*, and report it as a summary statistic of each grammar. For the experiments conducted here, the majority of grammars have over 90% coverage; see appendix B for more details. Note that due to the sampling procedure, the distribution of lengths between positive and negative samples is not matched; see fig. 9 for the ratios of positive to negative examples for each length $1 \leq \ell \leq 50$. In our analyses (§5.2), we rebalance the classes.

Data novelty & contamination. RELIC is designed to mitigate issues of data contamination by synthetically generating novel grammars rather than relying on a fixed dataset. Since the number of possible grammar-string pairs grows very rapidly in both the size of the grammar and the length of the string (see appendix A for more details), the risk of ever sampling a new grammar which could have been seen by a model during training is vanishingly small.

Intended use. RELIC is designed as a *zero-shot* evaluation of a model's ability to reason about complex in-context tasks *without also using positive and negative exemplars*. Evaluating models in a 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—for example, ones that rely on differences in the n-gram distributions in positive versus negative examples.

[3]Our codebase also includes the ability to sample negative strings adversarially by making minimal edits to positive samples. Since model performance on RELIC-500 is quite poor across the board, in our experiments we only report our results for the easier class of negative examples sampled from the unigram distribution, but for the purpose of evaluating more advanced models in the future, we encourage the use of the adversarial negative examples.

Dataset & code release. For reproducibility and to facilitate the creation of new RELIC benchmarks, we release RELIC-500 and the codebase to generate new grammars and examples at `https://jpetty.org/relic`.

4 Experimental Methodology

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` (OpenAI 2025b). Second, we evaluate two reasoning models from OpenAI, `o4-mini` and `o3` (OpenAI, 2025c) which are trained to generate a variable number of hidden tokens between the prompt and the final output. 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-weight instruction-tuned language models (`gemma-3-1b-it` and `gemma-3-4b-it`; Gemma Team et al. 2025). Finally, we evaluate an open-weight reasoning model, `DeepSeek-R1-Distill-Qwen-7B` (DeepSeek-AI et al., 2025). See appendix D for more details on API costs, compute, and evaluation hyperparameters.

For each grammar **G** and each string **s**, we prompt the model with a description of the language recognition task, the grammar, and the string, and ask the model to classify the string according to whether or not it is generated by the grammar, as shown below.[4]

PROMPT 1

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
```

[4]Prompts and LLM responses are shortened for brevity and clarity by eliding ancillary text with [...].

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

```
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'.

We use a regular expression to determine 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 (instead of 10) due to the increased cost of running evaluations on these models.

5 Experimental Results

5.1 Overall Performance

In Figure 2, we report models' mean class-balanced accuracy and their macro F1 score (mean of the per-class F1 score); since models do not have access to the distribution from which positive and negative examples are drawn, class-balancing penalizes models for incorrect biases towards predicting one class (positive or negative) over another. Indeed, models' accuracy often differs significantly between positive and negative examples, indicating such a bias: `gpt-4.1-nano` and `gpt-4.1-mini` have a much higher accuracy on positive examples than they do on negative ones, while the reverse is true for `gpt-4.1`, `o4-mini`, and `o3` (fig. 4, top).

This difference in accuracy is mostly attributable to the individual models' bias towards one of the two classes (fig. 4, bottom); while most models exhibit a strong tendency to classify short examples as negative, `gpt-4.1-nano` and `gpt-4.1-mini` increasingly classify examples as positive as they get longer, to the point that nearly all their predictions for examples of length $\ell = 50$ are positive. `gpt-4.1`, by contrast, exhibits nearly the opposite tendency, classifying most examples as negative across all lengths.

Model	Accuracy (%)	Macro F1
`gpt-4.1-nano`	54.1 ± 3.0	44.4 ± 0.7
`gpt-4.1-mini`	63.4 ± 1.9	57.5 ± 1.3
`gpt-4.1`	53.9 ± 4.5	48.5 ± 0.9
`o4-mini`	59.2 ± 3.5	58.1 ± 1.0
`o3`	70.4 ± 1.9	70.1 ± 1.1
`gemma-3-1b`	48.8 ± 2.3	30.9 ± 0.3
`gemma-3-4b`	48.7 ± 2.1	34.3 ± 0.7
`DSR1-7B`	47.9 ± 3.3	29.6 ± 0.3

Figure 2: Overall performance (class-balanced accuracy and macro F1 score) on RELIC-500. Chance performance is 50%. Performance is calculated first by grouping observations by grammar, example type, and example length, and then taking the mean. Errors reflect the standard error of the mean.

5.2 Performance Decreases as Grammar and Example Complexity Increases

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 nonlexical 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 nonlexical 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. 3, 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. 3, bottom). This drop-off happens quite rapidly, with an inflection point occurring 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 2 and 3 in appendix B).

5.3 Models Generally Agree on Which Grammars and Examples are Hard

Alongside the strong correlation between accuracy and grammar complexity, we found substantial variability in accuracy among grammars of similar

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

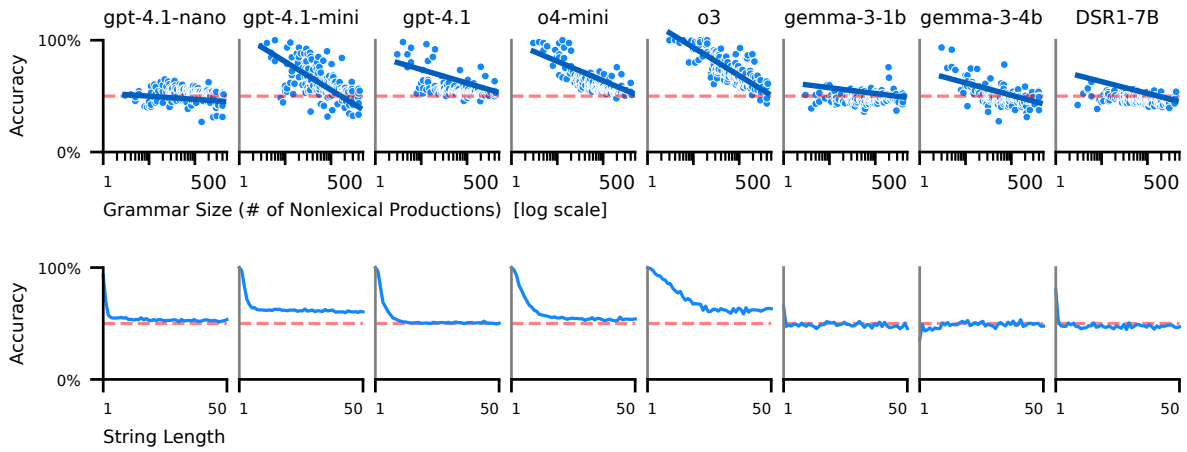

Figure 3: Models' accuracy on RELIC-500 reduces to near-chance (dashed lines) for all models as grammar size (number of nonlexical productions in the grammar, **top row**) and string length (**bottom row**) increase.

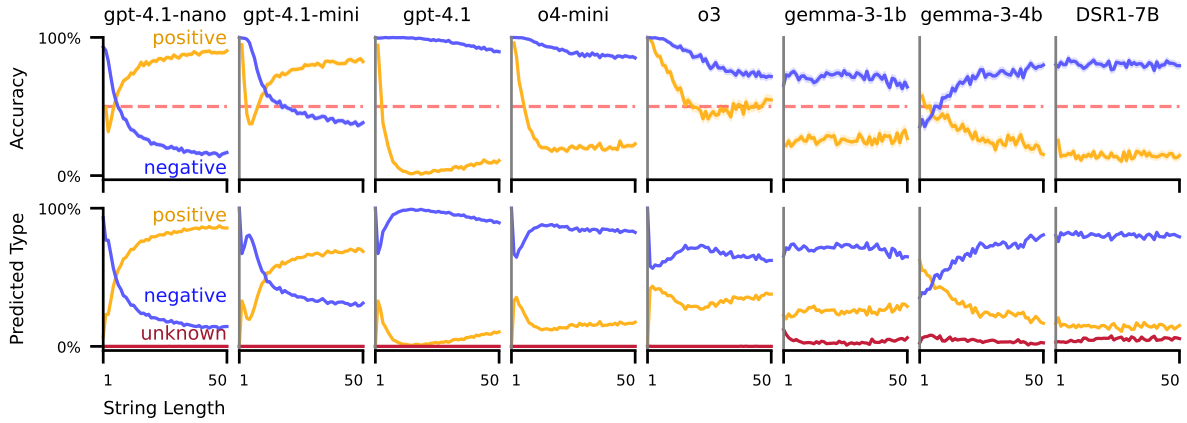

Figure 4: Accuracy on positive examples (that is, examples that can be generated by the grammar in question) and negative examples differs between models and across string lengths (**top row**); this is partially attributable to models' biases towards predicting one class (positive or negative) over another (**bottom row**).

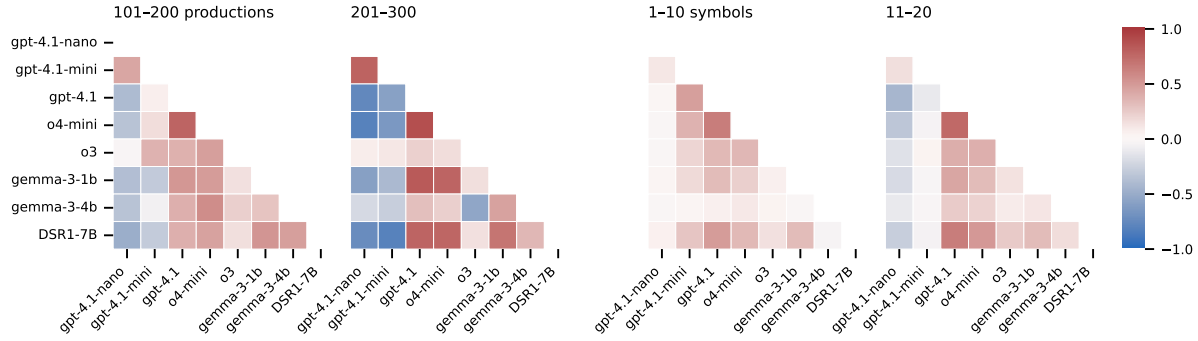

Figure 5: Spearman's rank correlation coefficients for the mean accuracy per grammar (**left**) and per example (**right**) between different models, binned by complexity. Most models agree on which grammars and examples are hard, with the exception of `gpt-4.1-nano/mini` which tend to disagree with other models. The strength of these correlations increases modestly as grammar and example complexity grow.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

complexity; this is particularly the case for the models with higher overall accuracy, `gpt-4.1-mini`, `o4-mini` and `o3` (fig. 3). Are the same grammars or examples challenging for all models? To test this, we rank the grammars and examples by their accuracy for each model and compute the Spearman's rank correlation across models. To mitigate the overall correlation with grammar complexity, we first divide the grammars into five bins of 100 production rules each; we similarly divide examples into length bins, in increments of 10. We find that grammars (fig. 5, left) and samples (fig. 5, right) have generally consistent difficulty across models. Of note is the fact that two models, `gpt-4.1-nano` and `gpt-4.1-mini`, are less correlated with the remaining models but have positive correlation with one another; this behavior likely reflects the fact that these models have a bias in favor of predicting samples to be positive, unlike the pattern of all other models as shown in fig. 4. We also observe that these (anti-)correlations strengthen as grammar and example complexity increase: correlations are closer to zero on grammars with 100–200 productions than they are for those with 200–300 productions, and similarly for examples of lengths 1–10 and 11–20. For a full overview of these correlations, see fig. 11 in appendix C.

6 How Do LLMs Succeed and Fail?

To understand how models attempt to solve RELIC, where they succeed, and why they fail, we examine the chains-of-thought produced by models prior to providing their final answers; we have access to intermediate outputs for all models except `o3` and `o4-mini`, whose CoTs are not shared with the user. We perform a qualitative analysis of model traces to identify high-level strategies models employ when responding. We then quantify how model strategy changes as a function of task complexity, and correlate these observations to changes in test-time compute expenditure.

We note that while theoretical (Merrill and Sabharwal, 2023) and empirical (Turpin et al., 2023; Pfau et al., 2024) work shows that CoTs may not be faithful representatives of underlying computation, such separation arises mostly in adversarial or otherwise-biased settings, where LLMs have been deliberately trained or inadvertently prompted to elicit unfaithful CoTs; Lanham et al. (2023) found that such problems do not often arise in non-adversarial settings. As our setup does not contain any means to introduce such bias (zero-shot prompts preclude introducing spurious correlations in the exemplars), we suspect that the risk of unfaithfulness here is low.

6.1 Qualitative Analysis of Strategy

We observe that models can identify a valid strategy to solve RELIC: using the rules of the grammar to construct a parse tree of the string, or showing (by exhaustion or proof) that one cannot exist. In some cases, with small grammars and short strings, models can often follow that strategy through to arrive at the correct answer. In other cases, models fail because they get distracted by irrelevant information, hallucinate success criteria, or second-guess their (correct) choice of strategy and attempt to solve the problem by other, invalid means.

Successful attempts. In some cases, models correctly identify positive samples by identifying the relevant production rules and deriving a parse tree. The response below highlights such an instance in which `gpt-4.1-mini` is able to exhaustively search for derivations compatible with the provided example; this search involves recursively applying the rule `NT5` → `NT0 NT5` three times, resulting in a full derivation for the 5-symbol string `t30 t24 t24 t23 t4`. The model also successfully ignores production rules irrelevant to the particular example.

LLM RESPONSE 2

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:

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

[...]

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

Failed attempts. Models often fail on more complicated versions of the task even when they are able to solve it in the simpler cases. These failures have many different causes. In some instances, models fail because they *accumulate errors*. Appendices E.1 and E.2 highlight such cases in which models describe a valid way to solve the problem but make invalid inferences from intermediate conclusions, become distracted by irrelevant information, or otherwise commit mistakes in the process of trying to follow a good strategy to its conclusion.

In other instances, models fail because they rely on *invalid strategies*. Although the nature of the problem is unchanged, prompting models with more complex grammars and strings can cause models to move away from valid, rule-based approaches and instead attempt to solve the task based on irrelevant or invalid heuristics, such as string length or the number of different ways to derive a single terminal symbol. Importantly, these heuristics can partially coincide with correct conclusions, leading to models being right for the wrong reasons (McCoy et al., 2019). Appendix E.3 shows just such a case, where a model rejects a string because it does not contain a particular symbol; while the response is correct, the reasoning is doubly invalid: not only does accepting the string not involve ensuring the presence of this symbol, but the symbol in question is not even in the grammar, and so its presence would be sufficient to reject the string.

6.2 Quantitative Analysis of Strategy

In light of the finding that models can propose valid solutions to RELIC but don’t always implement those solutions, we perform a quantitative analysis to explore how model strategy changes as a function of task complexity. We employ the “LLM-as-a-judge” framework (Zheng et al., 2023), prompting `gpt-5` to classify model completions as either *rule-based* (requiring a full derivation of the string, or a proof that one cannot exist) or *heuristic* (appealing to properties of the example or grammar to argue why an example seems likely or unlikely to be derivable). We focus on the `gpt-4.1` series of models, since they can solve the task in simple cases, fail on complex ones, and provide full access to their CoTs. We perform human and LLM validation of the LLM-judge; see appendix F for details.

Figure 6 (bottom) shows that model strategy is strongly dependent on task complexity. When plotted against string length, the proportion of responses employing a rule-based approach peaks very early and then falls off quickly as all three models in the `gpt-4.1` series shift to using invalid heuristics to arrive at an answer.

6.3 Test-Time Compute Grows Insufficiently

Models can scale their test-time compute (TTC) expenditure by producing more intermediate CoT tokens in a bid to solve harder problems. We examine how models expend TTC on RELIC by exploring how the length of the candidate strings impacts the number of tokens a model produces during inference (relative to the maximum token limit set for the model). Solving RELIC requires transformer LLMs to increase compute expenditure superlinearly in string length (§2.1). Yet fig. 6 (top) shows that relative TTC grows *sublinearly*, peaks early at levels far below ($\sim$ 10%) the token limits set for each model, and then *decreases* as a function of string length. Moreover, the point at which TTC

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

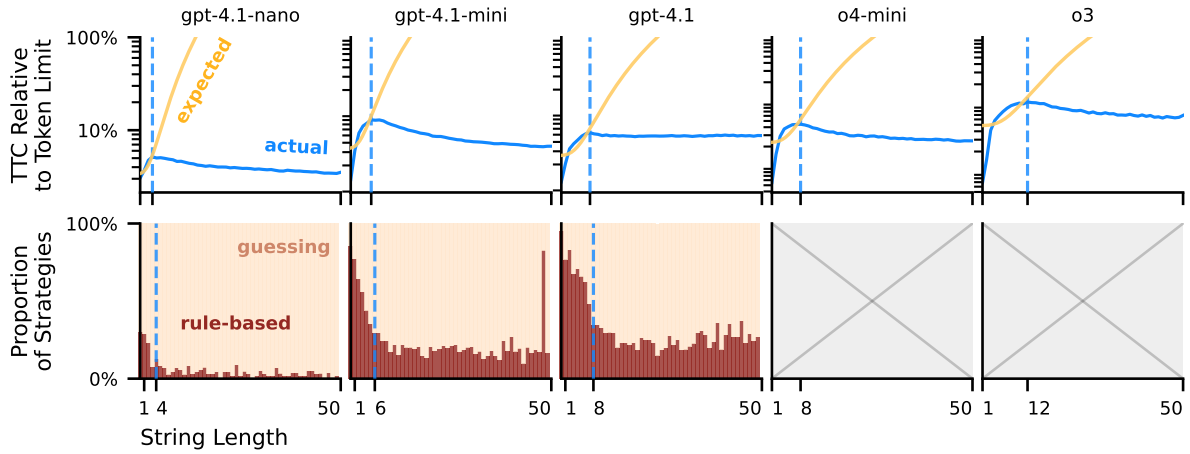

Figure 6: As string length increases, the test-time compute (TTC) expended by models peaks early and then diminishes (**top row**, blue line); this contrasts with the cubic growth that would be expected from applying the correct algorithm (yellow line). The cubic curve is fit to points before the peak, and TTC is computed as the mean number of completion tokens produced for examples of a given length, relative to the model's maximum token limit. As relative TTC stops increasing, models shift from pursuing rule-based approaches to guessing (**bottom row**, based on `gpt-5`'s classifications of the models' chains-of-thought). Neither `o4-mini` nor `o3` provide full CoTs, so we cannot classify their strategies directly, but the similarity in compute expenditure to models that report CoTs suggests a similar shift in strategy.

peaks and starts to drop off occurs close to the peak and drop-off in preference for rule-based strategies over heuristic guesses, further supporting the hypothesis that the models stop attempting to apply the correct strategy once the string reaches a certain length.

7 Finetuning on RELIC Does Not Improve Performance

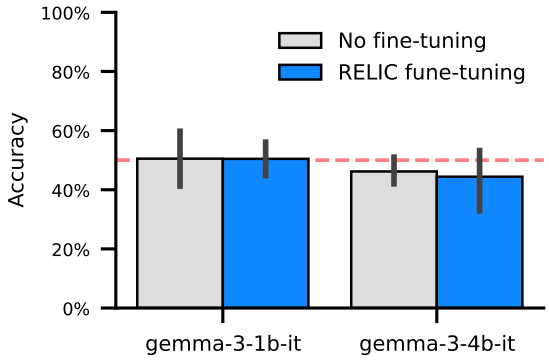

Figure 7: Finetuning the Gemma 3 models directly on RELIC grammar–string–label data does not confer any statistically significant benefit for the 1B or 4B models; error bars report 95% confidence intervals for mean model accuracy over five random seeds.

Could the weaker performance of smaller, open-weight models we study be due to a limited ability to infer the task from the prompt, as opposed to weak reasoning capabilities? To test this possibility, we directly tune the two Gemma 3 variants studied here (`-1b-it` and `-4b-it`) on a subset of RELIC-500. Since these two models are fine-tuned to be chat models, we format the fine-tuning samples using the Gemma chat templates for multi-turn conversations, with the `user` turn being the standard RELIC prompt used during evaluation and the `model` turn being the correct answer. See appendix G for details on fine-tuning.

Figure 7 shows that directly fine-tuning the Gemma 3 models on grammar–string–label data has no statistically significant impact on model performance: for both the 1B and 4B models, mean validation accuracy after fine-tuning remained within the 95% confidence intervals of the mean accuracies of the non-fine-tuned models. This suggests that these models' poor performance is not primarily due to inability to infer the task that needs to be performed from a single prompt.

8 Discussion

8.1 LLMs Quiet Quit on Hard Tasks

Across a wide variety of contemporary LLMs, models often demonstrate a good high-level understanding of the RELIC task: models can correctly outline sound strategies to solve in-context language

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

recognition and apply these strategies on simple instances of the task. Yet increasing the difficulty of RELIC by making strings longer and grammars larger—though still considerably smaller than the grammars of natural languages—induces a significant degradation in performance, even among the most capable reasoning models trained to "think longer" for harder problems. Quantitative analysis of their token expenditures shows that models stop increasing their inference compute expenditure as a function of task complexity quite early on, in contrast to theoretical requirements for maintaining accuracy on RELIC. Instead, compute expenditure peaks early, well short of the models' allotted compute budgets, and then *decreases*. Qualitative analysis on models where reasoning traces are available reveals that this shift in compute expenditure accompanies a dramatic reversal in reasoning *strategy:* as problems become harder, models are increasingly less inclined to apply sound reasoning strategies, instead opting to rely on shallow guesses.

We observe the same trend of diminishing TTC expenditure as a function of task complexity in the reasoning models for which we do not have access to reasoning traces (here, `o4-mini` and `o3`) as we do in those with visible intermediate reasoning (e.g., the `gpt-4.1` series). Since the computational complexity of RELIC requires that models grow their compute expenditure as a function of task difficulty, the observed behavior means that these reasoning models must be exhibiting the same shift in reasoning strategy which can be directly observed in the other models: as the task becomes harder, they start to guess. This results in a peculiar phenomenon where the best reasoning models must be "quiet quitting" on hard tasks which they could succeed at if they applied better strategies.

8.2 RELIC Predicts Reasoning Performance

CFL recognition may be hard for LLMs not only because of its computational complexity (§2.1) and context use requirements (§2.2), but also because the task itself is out-of-distribution and involves using data presented in a particular format that may be under-represented in training. To validate that success on RELIC is not merely an artifact of the LLMs' inability to infer the task itself, we compare performance on RELIC to that on a variety of other benchmarks: reasoning and knowledge benchmarks MMLU Pro (Wang et al., 2024) and GPQA Diamond (Rein et al., 2024); IFEval (Zhou et al., 2023), an instruction following benchmark; and IFEvalCode (Yang et al., 2025), an instruction-following benchmark specifically focusing on code generation. Figure 8 shows that performance on RELIC (as measured by the maximum length of strings that models can reliably classify above-chance) broadly correlates with performance on these more general-purpose benchmarks, suggesting that success or failure on RELIC is not merely an artifact of a model's ability to understand the task formatting but is more broadly correlated with its general-purpose utility on reasoning tasks.

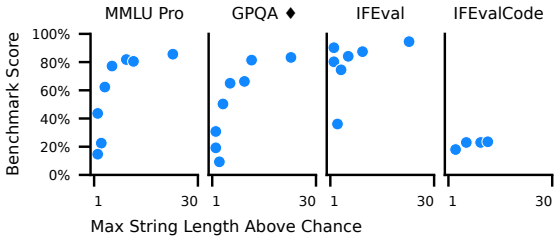

Figure 8: Performance on RELIC (here, measured by the longest string lengths for which models are reliably above their minimum accuracy, which for most models is at-chance) generally correlates with reasoning and instruction following evaluations.

8.3 Scalably-Complex Tasks Can Evaluate Hidden Reasoning

Proprietary reasoning models, such as `o4-mini` and `o3`, commonly obscure their reasoning traces, presenting the user with only their final response and potentially a summary of the hidden traces, along with metadata about how many tokens of compute were used to arrive at that final answer. The fact that these traces are withheld poses challenges for evaluation of such models, since it is impossible to conduct any qualitative analysis on the responses to determine whether the reasoning strategies employed are sound or consistent with a user's guidance.

Scalably-complex tasks like RELIC offer a partial window into the reasoning traces of these proprietary models. By connecting the intrinsic computational complexity of the task, here measured by the time-complexity of the best algorithms for solving RELIC as a function of its generating hyperparameters, to the computational behavior of language models, we can derive predictions for how test-time compute budgets *must* increase as a

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

function of task complexity; comparing this predicted behavior to what is observed empirically reveals whether closed-reasoning models are, in general, applying sound reasoning strategies.

8.4 Models Struggle at Dense Context Use

Models' poor performance on RELIC highlights that difficulties in context use can arise from a prompt's *density* in addition to its length. The grammars and strings used in RELIC are small compared to the models' context windows, amounting to no more than a few hundred unary or binary production rules and strings of no more than 50 words. Despite this, the highly-compositional nature of the task presents challenges for current LLMs: since it is not *prima facie* obvious which nonterminal rules must be retrieved from the context, nor the order in which they must be combined, RELIC requires models to effectively search over combinations of pieces of context to arrive at an answer, making the task much harder than many long-context use evaluations such as multi-hop needles or state evaluation benchmarks like NeedleBench (Li et al., 2025) or Michelangelo (Vodrahalli et al., 2024).

8.5 Limitations

RELIC does not control for task familiarity. While RELIC's nature as a synthetic benchmark means we can be very confident that any particular evaluation sample is novel for an LLM, it is quite possible that different models have been exposed to the concept of CFL recognition and the associated solutions. As such, success on RELIC likely depends on models having some prior understanding of the problem setup and does not constitute evidence that models are able to develop novel *solutions* to the task, but merely that they may be able to *apply* known solutions to novel instances of the task.

Strategy classification is subject to error. While our LLM-judge attained reasonable levels of agreement with human annotators on whether models were guessing versus attempting to employ good reasoning strategies when solving RELIC, the ability to further verify whether or not models make low-level mistakes in their chains of thought is not here addressed at scale. While we do observe instances of this, such as in the response shown in appendix E.3 where a model hallucinates a success condition based on a terminal symbol which doesn't appear in the grammar, we do not know how often such cases appear. Since these cases are difficult to automatically detect and verify, we think it possible that the LLM-judge strategy classifications may be overly rosy, as many instances classified by the LLMs as displaying rule-based reasoning may contain errors which make the arguments invalid or unsound.

8.6 Future Directions

The results here suggest several avenues of future research. First, the negative results of our fine-tuning experiments, albeit on small models, imply that different strategies must be pursued to improve model performance. In particular, it may be the case that training models on more structured completions involving guided reasoning strategies, such as full derivations or exhaustive searches, is more effective than merely tuning on labeled outputs (Lambert et al., 2025). Likewise, using in-context samples to generalize to novel grammars may be a promising way to attain better fidelity to rule-based approaches. Explicit guidance on which particular strategies are to be used may further boost performance, though at the cost of requiring the user to do upfront work to identify what those solutions are. Taken as a proxy for more general reasoning tasks, it is more desirable to find ways of improving performance which do not require the user to proactively identify valid completion strategies on a per-prompt basis.

Second, extending the class of grammars surveyed here to include more complex languages, such as context-sensitive languages, may yield further insight into how LLMs fare on tasks that are computationally harder than what is tested here; extension to context-sensitive languages would provide a clear analogy to the task of determining the syntactic validity of programs in many programming languages, which have non-context free components such as macros in C.

Finally, it may be possible to leverage a similar experimental setup to test for LLMs' ability to reason about formal languages in tasks other than language recognition; a promising candidate is to explore whether or not reasoning models can use formal grammars as a means of *translating* between two formal languages when given their specifications. Such an evaluation would provide a way to estimate how well LLMs can do in-context translation of natural languages (Tanzer et al., 2023; Petty et al., 2026).

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

Acknowledgments

Thanks to the NYU Computation and Psycholinguistics Lab for discussion, and to the anonymous reviewers for their valuable feedback. This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. This project is supported by the National Science Foundation (NSF) under grant NRT-HDR: FUTURE as well as Grant No. IIS-2239862.

References

Anthropic. 2025. Claude's extended thinking. https://www.anthropic.com/news/visible-extended-thinking. Accessed: 2025-12-5.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Re. 2024. Zoology: Measuring and Improving Recall in Efficient Language Models. In *The Twelfth International Conference on Learning Representations*.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. 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. Association for Computational Linguistics.

Bernd Bohnet, Kevin Swersky, Rosanne Liu, Pranjal Awasthi, Azade Nova, Javier Snaider, Hanie Sedghi, Aaron T Parisi, Michael Collins, Angeliki Lazaridou, Orhan Firat, and Noah Fiedel. 2024. Long-span question-answering: Automatic question generation and QA-system ranking via side-by-side evaluation. *Preprint*, arXiv:2406.00179.

T L Booth and R A Thompson. 1973. Applying probability measures to abstract languages. *IEEE transactions on computers. Institute of Electrical and Electronics Engineers*, C-22(5):442–450.

Alexandra Butoi, Ghazal Khalighinejad, Anej Svete, Josef Valvoda, Ryan Cotterell, and Brian DuSell. 2025. Training Neural Networks as Recognizers of Formal Languages. In *The Thirteenth International Conference on Learning Representations*.

N Chomsky and M P Schützenberger. 1963. The Algebraic Theory of Context-Free Languages. In P Braffort and D Hirschberg, editors, *Studies in Logic and the Foundations of Mathematics*, volume 35 of *Computer Programming and Formal Systems*, pages 118–161. Elsevier.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and control*, 2(2):137–167.

Alexander Clark. 2017. Testing distributional properties of context-free grammars. In *Proceedings of The 13th International Conference on Grammatical Inference*, volume 57 of *Proceedings of Machine Learning Research*, pages 42–53, Delft, The Netherlands. PMLR.

Alexander Clark. 2018. syntheticpcfg: Code for generating synthetic PCFGs for testing grammatical inference algorithms.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z F Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. Technical report, DeepSeek AI.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A Ortega. 2023. Neural Networks and the Chomsky Hierarchy. In *The Eleventh International Conference on Learning Representations*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and Fate: Limits of Transformers on Compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

Matthew Finlayson, Kyle Richardson, Ashish Sabharwal, and Peter Clark. 2022. What makes instruction learning hard? An investigation and a new challenge in a synthetic environment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 414–426, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 Technical Report. Technical report, Google DeepMind.

Google. 2025. Gemini 2.5: Our most intelligent AI model. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/ Accessed: 2025-11-10.

Raymond Greenlaw, Walter L Ruzzo, and James Hoover. 1991. A compendium of problems complete for P (preliminary). Technical report, University of Alberta.

Kavi Gupta, Kate Sanders, and Armando Solar-Lezama. 2025. Randomly sampled language reasoning problems reveal limits of LLMs. *Preprint*, arXiv:2501.02825.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the Real Context Size of Your Long-Context Language Models? In *First Conference on Language Modeling*.

Greg Kamradt. 2023. gkamradt/LLMTest_NeedleInAHaystack. https://github.com/gkamradt/LLMTest_NeedleInAHaystack/blob/main/README.md. Accessed: 2025-4-10.

Najoung Kim and Tal Linzen. 2020. COGS: A Compositional Generalization Challenge Based on Semantic Interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 9087–9105, Online. Association for Computational Linguistics.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing Frontiers in Open Language Model Post-Training. In *Second Conference on Language Modeling*.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. Measuring faithfulness in Chain-of-Thought reasoning. *Preprint*, arXiv:2307.13702.

Lillian Lee. 2002. Fast context-free grammar parsing requires fast boolean matrix multiplication. *Journal of the ACM*, 49(1):1–15.

Mo Li, Songyang Zhang, Taolin Zhang, Haodong Duan, Yunxin Liu, and Kai Chen. 2025. NeedleBench: Evaluating LLM Retrieval and Reasoning Across Varying Information Densities. *Transactions on Machine Learning Research*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Stroudsburg, PA, USA. Association for Computational Linguistics.

William Merrill and Ashish Sabharwal. 2023. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545.

William Merrill and Ashish Sabharwal. 2024. The Expressive Power of Transformers with Chain of Thought. In *The Twelfth International Conference on Learning Representations*.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024. OpenAI o1 System Card. *arXiv [cs.AI]*, arXiv:2412.16720.

OpenAI. 2024. Learning to reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2025-5-9.

OpenAI. 2025a. GPT-5 System Card. Technical report, OpenAI.

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

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

Jackson Petty, Jaulie Goe, and Tal Linzen. 2026. Evaluating in-context translation with synchronous context-free grammar transduction. *Preprint*, arXiv:2604.07320.

Jacob Pfau, William Merrill, and Samuel R Bowman. 2024. Let's Think Dot by Dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. 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. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. In *First Conference on Language Modeling*.

Walter L Ruzzo. 1980. Tree-size bounded alternation. *Journal of computer and system sciences*, 21(2):218–235.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V Nayak, Debajyoti Datta, and 21 others. 2022. 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.

Parshin Shojaee, Seyed Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Garrett Tanzer, Mirac Suzgun, Eline Visser, Dan Jurafsky, and Luke Melas-Kyriazi. 2023. A Benchmark for Learning to Translate a New Language from One Grammar Book. In *The Twelfth International Conference on Learning Representations*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R Bowman. 2023. Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*.

H Venkateswaran. 1991. Properties that characterize LOGCFL. *Journal of computer and system sciences*, 43(2):380–404.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, Rohan Anil, Ethan Dyer, Siamak Shakeri, Roopali Vij, Harsh Mehta, Vinay Ramasesh, Quoc Le, Ed Chi, Yifeng Lu, and 5 others. 2024. Michelangelo: Long context evaluations beyond haystacks via Latent Structure Queries. *Preprint*, arXiv:2409.12640.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

Quentin Gallouédec. 2020. TRL: Transformers Reinforcement Learning.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022a. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. 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. Association for Computational Linguistics.

Yiwei Wu, Atticus Geiger, and Raphaël Millière. 2025. How do transformers learn variable binding in symbolic programs? *Preprint*, arXiv:2505.20896.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large Language Models as Optimizers. In *The Twelfth International Conference on Learning Representations*.

Jian Yang, Wei Zhang, Shukai Liu, Linzheng Chai, Yingshui Tan, Jiaheng Liu, Ge Zhang, Wangchunshu Zhou, Guanglin Niu, Zhoujun Li, Binyuan Hui, and Junyang Lin. 2025. IFEval-Code: Controlled Code Generation. *Preprint*, arXiv:2507.22462.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *Advances in Neural Information Processing Systems*, 36:11809–11822.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for Large Language Models. *Preprint*, arXiv:2311.07911.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *The Twelfth International Conference on Learning Representations*.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

A Grammar and String Novelty

The grammars and candidate strings generated by RELIC are almost certainly novel, meaning that they are very unlikely to have been generated and observed before during training.

Lemma A.1. *Let G be a context-free grammar in Chomsky normal form with n nonterminal symbols and $|\Sigma| = t$ terminal symbols, and let ℓ be the length of a string drawn from Σ^+. Then there are at least $2^{n^3+nt-2n}$ many distinct grammars and t^ℓ many distinct strings.*

Proof. To establish a lower bound on the number of reduced CFGs in Chomsky normal form we exhibit a set of such grammars and count them. Fix $V = \{\mathtt{NT}_1, \ldots, \mathtt{NT}_n\}$ and $\Sigma = \{\mathtt{t}_1, \ldots, \mathtt{t}_t\}$. Let G' be a base grammar whose nonlexical productions are all rules of the form

$$\{S \to \mathtt{NT}_i \ \ \mathtt{NT}_1\} \quad \text{for } 1 \le i \le n, \qquad (\alpha)$$

and whose lexical productions are all rules of the form

$$\{\mathtt{NT}_i \to \text{`}\mathtt{t}_1\text{'}\} \quad \text{for } 1 \le i \le n. \qquad (\beta)$$

It is clear that G' is reduced, since every nonterminal $\mathtt{NT}_i$ is accessible from the ith nonlexical rule by (α) and productive in the ith lexical rule by (β). Furthermore, G' remains reduced after adding any other lexical or nonlexical production rules since their inclusion does not affect productivity or accessibility. A lower bound on the number of reduced CNF CFGs is then the number of distinct grammars which may extend G'. There are n^3 possible distinct nonterminal production rules and nt possible distinct terminal production rules. Since G' contains $2n$ production rules, there are $n^3 + nt - 2n$ remaining distinct production rules to choose from, and so there are at least $2^{n^3+nt-2n}$ distinct reduced CNF CFGs.

The number of distinct strings on t symbols of length ℓ is trivial. ■

Remark. *The number of distinct grammar–sample pairs for fixed n, t, and ℓ is multiplicative:*

$$N_{pairs} \ge 2^{n^3+nt-2n} \cdot t^\ell.$$

This grows very *fast. In practice, the space of distinct grammars and samples is extraordinarily large, and the chance of accidentally generating a grammar–sample pair which had been observed during training is vanishingly small.*

B Dataset Information

B.1 Distributional Statistics for Grammars and Strings

To construct the static evaluation set used in our experiments, we first over-generate grammars (∼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. Table 1 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.

	n_{term}	n_{nonterm}	n_{lex}
n_{nonterm}	0.01		
n_{lex}	0.54	0.07	
n_{nonlex}	0.00	0.31	−0.01

Table 1: Correlations between generating hyperparameters for the released static set. Note that n_{lex} and n_{term} are inherently correlated.

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 nonterminal 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 $1 \le \ell \le 50$ 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. 9 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 rebal-

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

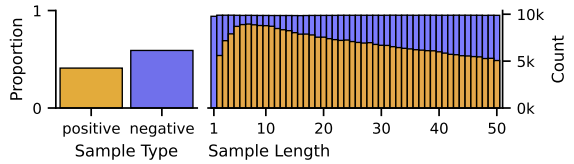

Figure 9: Proportions of example types represented in the static dataset, in aggregate (**left**) and broken down by example length (**right**).

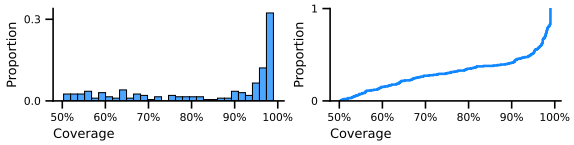

Figure 10: Distribution of grammars by coverage (i.e., the size of the language they generate, measured as the number of positive examples of lengths $\ell \leq 50$, with a maximum of 10 examples/length, out of a theoretical maximum of 500) shown as a histogram (**top**) and a cumulative distribution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

ance 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 number of examples for a grammar, 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. 10 shows the coverage of the grammars used in our experiments.

B.2 Why are n_{term} and n_{lex} Correlated?

In any context-free grammar the number of terminal symbols n_{term} is bounded above by the number of lexical production rules n_{lex}; to see why, consider the following minimal grammar with a single lexical rule and a single terminal symbol:

```
S → A A
A → 'a'
```

To add a new terminal symbol `'b'` to the grammar, we must add a production rule which yields it:

```
S → A A
A → 'a'
A → 'b'
```

Thus, $n_{\text{term}} \leq n_{\text{lex}}$. To see how it could be lower, consider a grammar like the following where multiple distinct nonterminals yield the same terminal:

```
S → A B
A → 'a'
B → 'a'
```

Here the grammar has two lexical production rules but only a single terminal symbol. A similar argument extends to the relationship between n_{nonterm} and n_{nonlex}.

C Regressions & Correlation Data

Table 2 reports the Pearson correlation coefficients for model accuracy and macro F1 score with the log transforms of 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).

We also report the coefficients of regression for accuracy (as a percentage) versus model type, n_{nonlex}, and example length ℓ in table 3. Both n_{nonlex} and ℓ were log-transformed and centered before being entered into the regression.

Figure 11 shows the full Spearman's rank correlation coefficients across models for the difficulties of individual grammars and samples.

D Inference Cost, Setup, and Hyperparameters

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

Table 4 reports the inference hyperparameters used in our experiments. For the open-weight 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.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

		$\log(n_{\text{term}})$	$\log(n_{\text{lex}})$	$\log(n_{\text{nonterm}})$	$\log(n_{\text{nonlex}})$
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 2: Pearson correlation coefficients r between models' accuracy/macro F1 scores and grammar hyperparameters, including the numbers of terminal symbols n_{term}, nonterminal symbols n_{nonterm}, lexical productions n_{lex}, and nonlexical productions n_{nonlex}. Correlation scores are taken over the mean F1 and accuracy values grouped by grammar.

E Chains-of-Thought

This appendix provides a few examples of the chains-of-thought (tokens generated between the prompt and the answer) produced by `DeepSeek-R1-Distill-Qwen-7B` (which we abbreviate here as `DSR1`).

E.1 Example A: Faulty Reasoning

In the example shown below, `DSR1` reports identifying a strategy that, while vaguely described, is consistent with the appropriate way to determine if the example is generated by the grammar (*"start from the string and see if I can break it down into the nonterminals according to the production rules"*; see ¶E.1). It begins exploring this strategy, but quickly backtracks and veers off into a heuristic and irrelevant analysis based on string length (*"Wait, let me count the tokens"*; ¶E.1). It then backtracks again, and attempts to find a derivation for the first symbol in the example (¶E.1). It correctly identifies two production rules that together can generate the first symbol in the example (the first symbol is `t75`, and the relevant rules are {`S` → `NT97 NT1`, `NT97` → `'t75'`}). It then proceeds to determine how the second symbol in the example, `t104`, could be derived. It notices that `NT1`—the second nonterminal in the rule `S` → `NT97 NT1`, which it used to derive the first symbol—does not directly produce `t104`, and considers this a potential reason for rejecting the string (¶E.1). It then considers the possibility that `NT1` produces `t104` not directly but through intermediate productions (¶E.1); it then rejects this possibility based on the presence of a different lexical production rule involving `NT1`—although this is an incorrect reason to reject this possibility—and concludes, incorrectly, that the example is not generated by the grammar (¶E.1).

LLM RESPONSE 3

... I think the approach is to start from the string and see if I can break it down into the nonterminals according to the production rules. ...

... 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. ...

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 nonterminals have a production ending with 't75'. Wait, in the

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

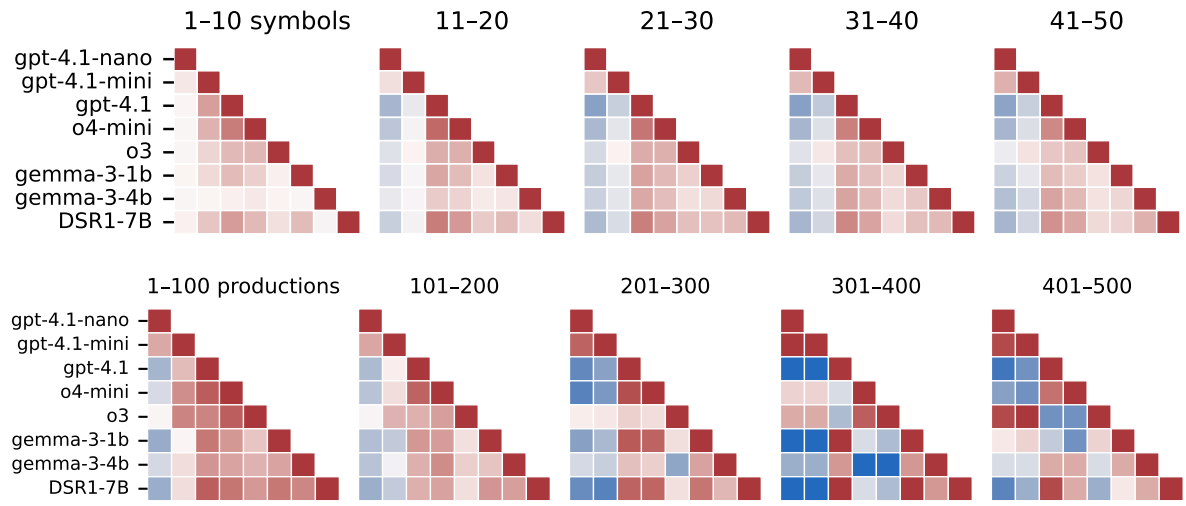

Figure 11: Spearman's rank correlation coefficients for accuracies between different models (**top:** per-example, faceted by example length; **bottom:** per-grammar, faceted by number of non-lexical productions).

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 3: 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 $p < 0.05$.

grammar, NT44 -> 't132', and NT197 -> 't178',

Model	Token limit	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 4: Inference hyperparameters used in experiments.

15

and NT97 -> 't75' according to the end of the grammar list: NT97 -> 't75'. So, NT97 can directly produce 't75'.

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.

Wait, alternatively, maybe the NT1 is not the immediate right child but another expansion. ...

Therefore, it's not possible that the entire string is generated by this grammar because the second symbol doesn't match 't1'. ...

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

E.2 Example B: Irrelevant Distractions

Here, DSR1 is prompted by a (negative) example consisting of *a single symbol* `t136`; though it identifies that this token is derivable by some lexical production rule, it immediately begins investigating irrelevant derivations which contain additional terminal symbols (¶1). It continues along this line before arguing (incorrectly) that the example seems plausible since it seems like there exist derivations containing the provided terminal symbol (¶2).

LLM RESPONSE 4

[...]

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

E.3 Example C: Incorrect Strategy

Here, DSR1 adopts a strategy that is entirely irrelevant: it decides to check if the string contains the symbol `t27` (¶3), a symbol which not only does not appear in the example it was given, but does not even appear in the grammar in question. It then proceeds to check if any symbol in the provided string is `t27` before rejecting the example because `t27` was not present (¶4). Because the example in this case happened to be a negative example, DSR1 happened to produce the correct answer, but it reached this answer for irrelevant reasons.

LLM RESPONSE 5

[...]

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'. ¶3

[...]

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'. ¶4

F Strategy Classification Details

To categorize the strategies used by the `gpt-4.1`-series of models, we employ the 'LLM-as-judge' framework (Zheng et al., 2023): we prompt a judge model to classify other models' responses according to a rubric. To validate this methodology, we collect human judgments on model strategies for 62 different completions, balanced for binned sample length (short, medium, and long) and sample class (positive or negative). Human judgments are provided by paper authors. We then select two candidate judge LLMs, `o4-mini` and `gpt-5`, and prompt these judge models to categorize the same 62 samples. We compare the majority-category selected by the human annotators to that of the LLMs to compute the validation agreement between human and LLM judgment. We then select the judge LLM with the highest validation agreement to classify model strategies for a much larger subsampling of the full data, and report these values in fig. 6. We prompt all judge models with the same guidelines given to human annotators, shown below:

PROMPT 6

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

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026

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}
```

We found that `o4-mini` had a human-agreement of 68% while `gpt-5` had a human-agreement of 83%, and so report `gpt-5`'s analysis in the main text, though we note that `o4-mini`'s classifications generally agree with those of `gpt-5`.

G Finetuning Details

We fine-tune `gemma-3-1b-it` and `gemma-3-4b-it` on a subset of RELIC (§7). We randomly select 100 grammars and generate an 80%-20% train-validation split, training on the RELIC prompt (see Prompt 1) in §4 along with the correct response. We fine-tune each model for five epochs, after which point validation loss begins to diverge. For each model size we perform five training runs from different random seeds. We use the AdamW optimizer (Loshchilov and Hutter, 2019) with standard parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and an initial learning rate of 5×10^{-5}, using the TRL framework from HuggingFace (von Werra et al., 2020).

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.735/2612412/tacl.a.735.pdf by guest on 11 July 2026