A FORMAL LANGUAGE BENCHMARK FOR LLMS*

Bishwamittra Ghosh[1], Krishna P. Gummadi[1], Evimaria Terzi[2]
[1]Max Planck Institute for Software Systems, Germany, [2]Boston University, USA

ABSTRACT

Empirical research has guided the progress of large language models (LLMs) over the years, where we often have a limited understanding of the underlying data fed to them. We take an orthogonal approach to the problem, and propose a *formal language* benchmark for studying LLMs.

We ask the following questions: (a) Why do we need formal language as a test bed to study LLMs?, and (b) How do we measure the language proficiency of an LLM? As contributions, we highlight the *preciseness* and *control* of probabilistic formal languages, which are well-suited for studying LLMs. Moreover, we make a contrast between a *generative* test and a *discriminative* test in determining the language proficiency of an LLM, where the latter is *comparable* across LLMs.

1 INTRODUCTION

The progress in large language models (LLMs) is continuously determined by how well they perform on a variety of benchmarks (Suzgun et al., 2023), such as common-sense reasoning (Trinh & Le, 2018), math solving (Mirzadeh et al., 2024), and text summarization (Zhang et al., 2024), etc. How well do we understand these benchmarks? Can we precisely define the learning tasks? Can we ensure that both training and test samples are in the same distribution? Can we guarantee that all LLMs are evaluated fairly and there is no risk of data contamination? Our paper aims to address these questions by proposing a *formal language* learning benchmark for LLMs.

Probabilistic formal language is a synthetic source of token-strings, identified by a probabilistic formal grammar (Chomsky, 1956). Formal language provides *preciseness* and *control*, which is hard to achieve in existing natural language datasets (Section 2). Specifically, formal language contains syntax only, where sampling is precise, i.e., all strings follow the grammar rules. Furthermore, formal language provides greater control in changing grammar rules, tokens, or entropy, which is needed to avoid data contamination in practice (Sainz et al., 2023).

A natural question we can ask is *how well does an autoregressive LLM learn a formal language*? Here, the learning task involves recognizing syntactic generalizable patterns of the language represented by training strings, by attempting to generate a new token from the prefix of the string (Bender et al., 2021). Once the language is learned, the LLM, due to its generative nature, can generate strings both inside and outside the language.

The paper, therefore, asks a foundational question: *how can we measure the language proficiency of the LLM*? Shall we consider (unseen) strings belonging to the language? What about strings that are close but outside the language? As a contribution, we make a contrast between a *generative* and a *discriminative* test for language proficiency (Section 3). We conclude that the discriminative test is comparable across LLMs and becomes more informative, where the LLM not only generates strings inside the language, but also generates them better than those outside the language. In contrast, the generative test only compares generation performance within the language (Bhattamishra et al., 2020), which is subjective to model-specific priors affecting results inadvertently.

Finally, we admit that syntax-focused formal language may not evaluate the full potential of LLMs that can reason with semantics. Instead, we invite the community to focus on precise and well-understood synthetic data, including formal languages, for a controlled scientific study of LLMs.

*Short paper

$S \to A16$ [1]
$A16 \to A15\ A13$ [0.50]
$A16 \to A13\ A15\ A14$ [0.50]
$A15 \to A12\ A11\ A10$ [0.50]
$A15 \to A11\ A12\ A10$ [0.50]
$A14 \to A11\ A10\ A12$ [0.50]
$A14 \to A10\ A11\ A12$ [0.50]
$A13 \to A11\ A12$ [0.50]
$A13 \to A12\ A11$ [0.50]
$A12 \to A8\ A9\ A7$ [0.50]
$A12 \to A9\ A7\ A8$ [0.50]
$A11 \to A8\ A7\ A9$ [0.50]
$A11 \to A7\ A8\ A9$ [0.50]
$A9 \to 9\ 8\ 7$ [0.50]
$A9 \to 8\ 7$ [0.50]
$A8 \to 6\ 5$ [0.50]
$A8 \to 6\ 4\ 5$ [0.50]
$A7 \to 3\ 1$ [0.50]
$A7 \to 1\ 2\ 3$ [0.50]

Figure 1: Inspired by Allen-Zhu & Li (2023), we illustrate an example hierarchical PCFG G_1, where non-terminals are makred in red, terminals are in teal, and rule-probabilities are in blue. The grammar contains non-terminal symbols S and A's, alphabet $\mathbf{T} = \{1, 2, \ldots, 9\}$, and probabilistic production rules which are applied in a hierarchical way. To generate a string, we start from the non-terminal S and recursively apply production rules until reaching terminals only.

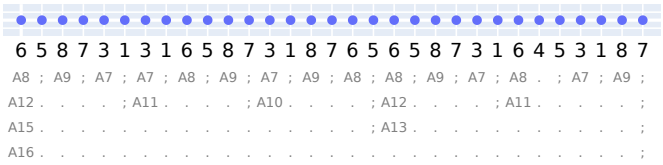


Figure 2: A string s from language L_1, generated by grammar G_1 in Figure 1. The rule '$A16 \to A15\ A13$ [1]' indicates that non-terminal $A16$ is expanded to $A15$ followed by $A13$ with probability 1, and so on, until reaching $\mathbf{T}$. The generation probability of s is the multiplication of the probabilities of rules applied recursively to generate s, and $P_{L_1}(s) = (0.5)^{23}$.

2 Formal Language Learning by LLMs

We propose a formal language learning setup for studying LLMs. Let an LLM M be trained on a dataset $D = \{s\}$ of strings, which are sampled from an underlying language. We specifically consider a *probabilistic formal language* L, which is defined on a set of allowed tokens, called alphabet, $\mathbf{T}$, and specifies a probability distribution P_L over strings, $P_L : \mathbf{T}^* \to [0, 1]$, and $\mathbf{T}^*$ is the set of all strings. The language is identified by a formal grammar[1], denoted by G, which contains terminals, non-terminals or alphabet, and probabilistic production rules – the grammar is an algorithm to generate strings. Find additional details in Appendix 8.

We consider probabilistic context-free grammars (PCFGs), where the production rules are applied in a hierarchy, similar to the recursive structure of natural language (Allen-Zhu & Li, 2023). In Figure 1, and 2, we show an example of PCFG, and a sampled string from the grammar, respectively. Below, we discuss the importance of preciseness and control of formal grammar based languages, which makes them suitable for studying LLMs.

Preciseness

- **Precise learning task.** In our setup, the training objective of the LLM is to generate the next token in the string, given a prefix of the string, which is the autoregressive learning task. This allows the LLM to recognize certain generalizable patterns of the language represented by the training strings, without seeing the exact grammar rules. During inference, the LLM can generate unseen strings, both from inside and outside the language. The probability (or cross-entropy loss) of generating new strings that are inside vs. outside the language determines whether the LLM is proficient in the language – we expand on this in Section 3.
- **Precise specification of a language.** We can verify whether a string generated by the LLM post-training belongs to the language or not, by performing a membership test of the string with the grammar. The membership test is tractable (polynomial in the length of the string) for the class of

[1]A finite language can indeed be identified by multiple grammars, whereas a grammar identifies a single language. In the paper, we consider a single grammar and see if the LLM can learn the corresponding language.

regular and context-free languages, and expensive for more expressive languages, such as context-sensitive and recursive languages. We focus on context-free languages in this paper.

- **Syntax-oriented specification.** Formal language contains symbols and rules, and the generated strings are defined by syntax. There is no notion of semantics, which may potentially lead to ambiguity. In fact, in a formal language, we can precisely specify which tokens the LLM should generate given a prefix of the string.
- **Precise Sampling.** Formal language allows us to sample training and test strings from the same distribution. In contrast, most existing datasets in LLM research often cannot guarantee that the training and test strings are sampled from the same distribution.

Control

- **Controlling grammar rules, alphabet, and entropy.** Formal grammars provide greater flexibility than existing natural language datasets. We can change the production rules to introduce new patterns, and change alphabet to communicate the same rules through a different set of tokens. Furthermore, we can change the probability of the rules to convert a high entropy language (as shown in Figure 1) to a low entropy language (as shown in Figure 6), where the accepted strings in the language are fixed, but their distribution differs.[2] Such a fine-grained control is missing in existing natural language datasets.
- **Synthetic data generation.** Formal language is a rich source of synthetic data, which helps to avoid the data contamination problem in LLM evaluation (i.e., one LLM performing better/differently than others due to having been trained on the data before).

Therefore, the preciseness and control of formal languages make them suitable for studying what the LLMs can learn once it is trained on a finite set of strings from the language.

3 Measuring Language Proficiency of LLMs

The central question in this section is how can we measure the language proficiency of an LLM? A critical desideratum is that when measuring language proficiency, do we only consider strings inside the language, or do we also consider strings outside the language? Based on this distinction, we discuss a generative test and a discriminative test for language proficiency, and show that the discriminative test is comparable across LLMs. Next, we state our research questions (RQs).

- **RQ 1 (Absolute Performance).** What is the absolute measure of language proficiency of an LLM M on a language L?
- **RQ 2 (Comparative Performance across Languages).** Between two languages L_1 and L_2 and an LLM M, in which language is M more proficient?
- **RQ 3 (Comparative Performance across LLMs).** Between two LLMs M_1 and M_2 and a language L, which LLM is more proficient on L?

We make a distinction between absolute and comparative performance, because there could be cases when the absolute value of language proficiency cannot be directly comparable across LLMs.

The generative Test. When we have access to strings inside the language, a straightforward test is to see how well the LLM generates strings in the language; higher the generation performance, better the language proficiency. Generative performance can be measured using probability, cross-entropy loss, or perplexity of generating a string, where higher probability and lower loss or perplexity are considered better. The simplicity of the test is adopted widely in the literature (Kallini et al., 2024; Jumelet & Zuidema, 2023; Bhattamishra et al., 2020; Wang, 2021; Akyürek et al., 2024).

Formally, let loss be a metric for generation performance. The *absolute* language proficiency of an LLM M on a language L is defined as the expected loss of generating all strings in L: $\texttt{loss}(M, L) = \mathbb{E}_{s\sim L}[\texttt{loss}(M, s)]$, where $\texttt{loss}(M, s) = -\frac{1}{|s|}\sum_{i=1}^{|s|} \log \Pr_M(s_i \mid s_{<i})$ is the loss (negative log likelihood) of generating all tokens in the string by the LLM. Thus, $\texttt{loss}(M, L)$ answers **RQ 1**.

[2]The entropy $H(L)$ of a language L is the entropy of the probability distribution of strings, $H(L) = -\sum_{s\in\mathbf{T}^*} P_L(s) \log P_L(s)$ (Cover, 1999; Carrasco, 1997).

Regarding *comparability*, M is more proficient in L_1 than L_2, if $\texttt{loss}(M, L_1) < \texttt{loss}(M, L_2)$, which answers **RQ 2**. On the other hand, M_1 is more proficient on L than M_2, if $\texttt{loss}(M_1, L) < \texttt{loss}(M_2, L)$, which answers to **RQ 3**. However, should we directly compare loss across LLMs?

Potential issues with the generative test. Generation loss is impacted by model priors. Two LLMs having different priors, such as being pre-trained on different datasets or having different model configurations (vocabulary or parameters), end up achieving non-comparable generation loss. For example, consider M_1 as monolingual and M_2 as multilingual, and both LLMs are trained optimally on L, i.e., they achieve respective lowest loss. Due to knowing multiple languages, M_2 may have higher loss on L than M_1. Does it mean that M_2 is less proficient than M_1? To this Extended, our argument is that when comparing across LLMs (**RQ 3**), generation loss should not be compared directly. What if we could access strings outside the language? Does it help us avoid this issue?

The Discriminative Test. The key intuition behind the discriminative test is: *if an LLM learned a language, it should generate strings in the language with lower loss than strings outside the language*. Thus, the discriminative test attempts to classify in-language and out-of-language strings based on their generation loss, where the success of classification is an implication of language proficiency. As shown in Figure 3, the test can be stricter by picking *close* out-language strings (according to some distance metric like edit distance) to in-language strings and checking if they can still be identified as out-of-language.

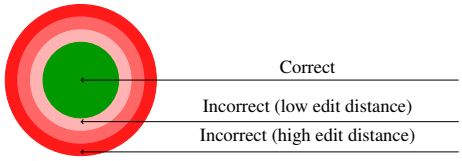


Figure 3: We visualize the set of all strings in a hierarchy, where the inner green circle denotes grammatically correct in-language strings, and the outer red circle denotes grammatically incorrect out-language strings. The generative test focuses on generation performance within the green circle, while the discriminative test focuses on comparative generation performance between green and red (specially at low edit distance) circles.

Formally, let $\mathsf{T}(L)$ denote out-language strings, constructed by editing or transforming strings in L and ensuring that they are not in L. Consider a binary (linear) classifier, where input is the generation loss of strings in $L \cup \mathsf{T}(L)$ by an LLM, and the classification task is to determine their membership. Let $\texttt{auc}_M(L, \mathsf{T}(L)) \in [0, 1]$ be the AUC (area under the receiver operating characteristic curve) of the classifier using model M; the higher the value the better.

Thus, the language proficiency of LLM M on language L is $\texttt{auc}_M(L, \mathsf{T}(L))$, answering **RQ 1** in the discriminative test. LLM M is more proficient in L_1 than L_2, if $\texttt{auc}_M(L_1, \mathsf{T}(L_1)) > \texttt{auc}_M(L_2, \mathsf{T}(L_2))$, answering **RQ 2**. Finally, LLM M_1 is more proficient on language L than LLM M_2, if $\texttt{auc}_{M_1}(L, \mathsf{T}(L)) > \texttt{auc}_{M_2}(L, \mathsf{T}(L))$, answering **RQ 3**.

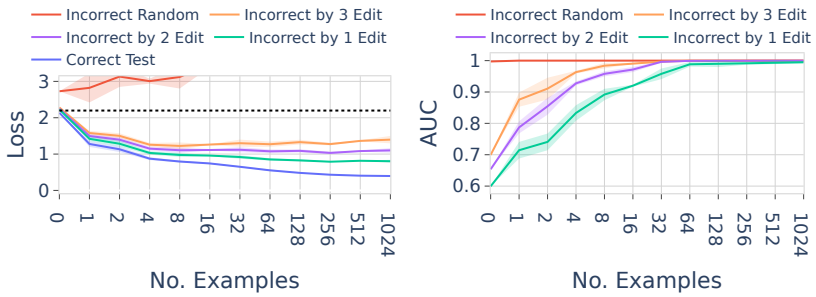
Discriminative test is comparable across LLMs. The discriminative test asks the same LLM to generate both in-language and out-of-language strings, and derive a classification score measuring how well the LLM can discriminate the strings in either side of the language boundary. Therefore, the derived classification score, which is normalized in $[0, 1]$, is comparable across LLMs.

The following analysis shows the difference between the two tests. Consider that M_1 generates strings in L with probability p, while M_2 generates with probability $p/2$. This can be achieved by creating a wrapper around M_1: if M_1 generates a string, M_2 generates the same string half of the times, and in the remaining times, it generates a random string. Certainly, M_1 is a better generator of L, but both are equally good discriminators. Similarly, referring back to our previous example of monolingual vs. multilingual LLMs, even if M_2 generates L worse than M_1, it can discriminate equally or perhaps better than M_1 (due to knowing multiple languages). Therefore, the classification score in the discriminative test, which is agnostic to model-priors, is comparable across LLMs.

4 Results

In Figure 4, we show the language proficiency of an LLM w.r.t. generative and discriminative tests.

Generative test alone is misleading. With increasing training examples, the loss decreases, i.e., generation is better, on in-language test strings, as well as strings that are close but outside the lan-



(a) Generative Test (b) Discriminative Test

Figure 4: Language proficiency of Mistral-7B on language L_1.

guage, such as grammatically incorrect strings by edit distance $k \in \{1, 2, 3\}$. Thus, *the generative test alone is insufficient in determining language proficiency on the* ***target*** *language.*

Discriminative test score is correlated with training size and the distance of out-language strings. The AUC of discriminating grammatically correct and incorrect strings increases with examples. In fact, AUC is correlated with the degree of incorrectness; higher the incorrectness, higher the AUC. *Importantly, at around* 128 *examples, the LLM achieves perfect discrimination (AUC* ≈ 1*) between close-distant test strings and incorrect strings by* 1 *edit – the strictest possible test.*

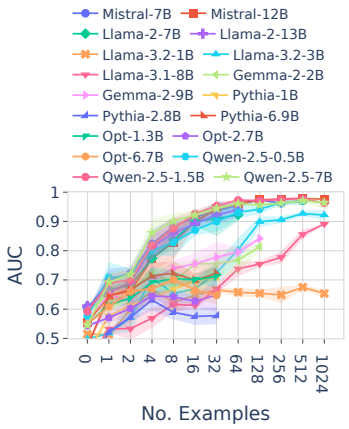


Figure 5: LLMs have variable language proficiency on learning language L_1.

AUC	Model
Good (≥ 0.75)	Qwen-2.5-7B, Mistral-7B, Qwen-2.5-1.5B, Llama-2-13B, Qwen-2.5-0.5B, Llama-2-7B, Mistral-12B
Moderate (≥ 0.6)	Gemma-2-2B, Gemma-2-9B, Pythia-6.9B, Opt-1.3B, Opt-6.7B, Pythia-1B, Llama-3.2-3B, Opt-2.7B, Llama-3.2-1B
Poor (< 0.6)	Llama-3.1-8B, Pythia-2.8B

Table 1: Ranking of different LLMs based on the median AUC up to 32 examples.

Different LLMs have variable language proficiency. We provide training strings as in-context examples and ask the LLM to generate new strings that are both inside and outside the language. In Figure 5, LLMs vary substantially in their language proficiency while learning the same formal language. Table 1 shows the relative ranking of different LLMs based on discriminative test scores. Thus, *formal language learning is not uniform across LLMs, and the discriminative test brings a more objective measure of language proficiency across different LLMs.*

5 Conclusion

We study LLMs' learning ability on formal languages. We discuss the preciseness and control of formal languages, which make them suitable for studying LLMs, in contrast to ill-understood natural language datasets. We propose a discriminative test to measure language proficiency, which is comparable across LLMs, and ranks them objectively on formal language learning. Future work includes studying in-depth on what aspects of the probabilistic language does an LLM learn. We aim to release a formal language benchmark for evaluating LLMs, inspired by the decade-long study of formal language theory.

REFERENCES

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Arhitectures and algorithms. *arXiv preprint arXiv:2401.12973*, 2024.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 1, learning hierarchical language structures. *ArXiv e-prints, abs/2305.13673, May*, 2023.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 610–623, 2021.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers to recognize formal languages. *arXiv preprint arXiv:2009.11264*, 2020.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Nadav Borenstein, Anej Svete, Robin Chan, Josef Valvoda, Franz Nowak, Isabelle Augenstein, Eleanor Chodroff, and Ryan Cotterell. What languages are easy to language-model? a perspective from learning probabilistic regular languages. *arXiv preprint arXiv:2406.04289*, 2024.

Rafael C. Carrasco. Accurate computation of the relative entropy between stochastic regular grammars. *RAIRO-Theoretical Informatics and Applications*, 31(5):437–444, 1997.

Ta-Chung Chi, Ting-Han Fan, Alexander I Rudnicky, and Peter J Ramadge. Transformer working memory enables regular language reasoning and natural language length extrapolation. *arXiv preprint arXiv:2305.03796*, 2023.

Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.

Michael Collins. Probabilistic context-free grammars (pcfgs). *Lecture Notes*, 2013.

Ryan Cotterell, Sabrina J Mielke, Jason Eisner, and Brian Roark. Are all languages equally hard to language-model? *arXiv preprint arXiv:1806.03743*, 2018.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, et al. Neural networks and the chomsky hierarchy. *arXiv preprint arXiv:2207.02098*, 2022.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Michael Hahn and Mark Rofin. Why are sensitive functions hard for transformers? *arXiv preprint arXiv:2402.09963*, 2024.

Thomas F Icard. Calibrating generative models: The probabilistic chomsky–schützenberger hierarchy. *Journal of Mathematical Psychology*, 95:102308, 2020.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Jaap Jumelet and Willem Zuidema. Transparency at the source: Evaluating and interpreting language models with access to the true distribution. *arXiv preprint arXiv:2310.14840*, 2023.

Julie Kallini, Isabel Papadimitriou, Richard Futrell, Kyle Mahowald, and Christopher Potts. Mission: Impossible language models. *arXiv preprint arXiv:2401.06416*, 2024.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

Sabrina J Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. What kind of language is hard to language-model? *arXiv preprint arXiv:1906.04726*, 2019.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D Manning. Characterizing intrinsic compositionality in transformers with tree projections. *arXiv preprint arXiv:2211.01288*, 2022.

Isabel Papadimitriou and Dan Jurafsky. Injecting structural hints: Using language models to study inductive biases in language learning. *arXiv preprint arXiv:2304.13060*, 2023.

Shauli Ravfogel, Yoav Goldberg, and Tal Linzen. Studying the inductive biases of rnns with synthetic variations of natural languages. *arXiv preprint arXiv:1903.06400*, 2019.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. Nlp evaluation in trouble: On the need to measure llm data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10776–10787, 2023.

Hui Shi, Sicun Gao, Yuandong Tian, Xinyun Chen, and Jishen Zhao. Learning bounded context-free-grammar via lstm and the transformer: difference and the explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 8267–8276, 2022.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. Transformers as Recognizers of Formal Languages: A Survey on Expressivity, October 2023. URL `http://arxiv.org/abs/2311.00208`. arXiv:2311.00208 [cs].

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024. URL `https://arxiv.org/abs/2403.08295`.

Trieu H Trinh and Quoc V Le. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*, 2018.

Shunjie Wang. *Evaluating transformer's ability to learn mildly context-sensitive languages*. University of Washington, 2021.

Jennifer C White and Ryan Cotterell. Examining the inductive bias of neural language models with artificial languages. *arXiv preprint arXiv:2106.01044*, 2021.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024.

6 Related Work

Many prior works have studied formal languages in the context of LLMs. There are two broader questions that most studies have asked.

What is the relative representation capability of LLMs compared to other sequences models, or more specifically, what classes of languages are learnable by an LLM? LLMs with a Transformer architecture may have a different representation capability than other neural language models (LMs) like LSTMs and RNNs. We refer to a recent survey discussing the expressiveness of LLMs as a language recognizer Strobl et al. (2023). Towards comparing representation capability, Shi et al. (2022) find that both LSTM and Transformer network can simulate CFL with bounded recursion having a similar representation power. However, LSTM has a disadvantage that it fails to decompose the latent representation space unlike a transformer. (Bhattamishra et al., 2020) observe a clear contrast between the performance of Transformers and LSTMs on regular languages. They find that in comparison with LSTMs, Transformers achieve limited performance on languages involving periodicity, modular counting, and even simpler star-free variants of Dyck-1 languages. Delétang et al. (2022) explore how neural network models used for program induction relate to the idealized computational models defined by the Chomsky hierarchy Chomsky (1956). They find that neural language models are hard to place on the standard Chomsky hierarchy. Several works criticize their setup, since they consider a language transduction task (mapping one language to another), which is different from the language recognition task Icard (2020). Borenstein et al. (2024) consider learning strings from deterministic and probabilistic finite state automata. They empirically test the learnability as function of various complexity parameters of the language and the hidden state size of the Transformer and RNN. In a different line of work, Akyürek et al. (2024) evaluate neural LM's abilities to learn regular languages in `ICL`. Rather than learning one particular distribution from the training dataset, they infer the generating mechanism using `ICL`. Similar to Delétang et al. (2022), they find that RNNs are better suited to modeling formal languages than Transformers. Kallini et al. (2024) construct a continuum of languages that differ in their hardness to learn and show that GPT-2, a variant of LLM, has difficulty in learning the carefully constructed impossible languages, compared to English.

While most of the works in this line capture the expressiveness of LLMs and its differing representation ability with other sequence models, one fundamental criticism we find is the evaluation metrics they consider. As elaborated in Section 3, they are focusing on testing how well an LLM learn the grammar rules or automata state, without utilizing the natural generation capability of the LLMs in generating strings from inside and outside the language. In contrast to their evaluation criteria, ours is more tailored towards how LLMs operate and become proficient in a language.

Does an LLM learn from a given distribution, if so how? Several studies utilize the controlled data generation of formal languages to study different NLP aspects of the LLM. Formal languages, particularly the one derived from context free grammars, can imitate the rich recursive structure of natural languages. Therefore, many studies focus on teaching the LLM strings from a formal language and explain how LLMs might learn them (Allen-Zhu & Li, 2023; Murty et al., 2022; Liu et al., 2022). In another line, Jumelet & Zuidema (2023) study if causal and masked LLMs capture the true underlying patterns if trained on a true distribution. They find that causal LLMs approximate the theoretically optimal perplexity of the PCFG more closely than masked LLMs. Along that direction, several studies consider the known distribution to analyze the impact of topological features of a language Cotterell et al. (2018); Mielke et al. (2019); Ravfogel et al. (2019); Mielke et al. (2019); Papadimitriou & Jurafsky (2023); White & Cotterell (2021). Several studies propose to augment additional component to LLMs to enable them learning certain class of languages with ease. For example, Chi et al. (2023) propose to add working memory, such as weight sharing, adaptive-Depth, and sliding-dilated attention to GPT model to enable it to learn parity function, which hard for an LLM to learn Hahn & Rofin (2024).

Our broader research goal is to design a formal language benchmarks, by accumulating formal grammars of different complexities, and evaluate LLMs on their learning ability.

7 EXPERIMENTAL SETUP

We experiment with 18 open-source LLMs from 6 families, such as Mistral (Jiang et al., 2023), Llama (Dubey et al., 2024), Qwen (Yang et al., 2024), Gemma (Team et al., 2024), Pythia (Biderman et al., 2023), and Opt (Zhang et al., 2022), ranging from 0.5B to 13B parameters. All reported results are averaged over three experimental runs. All experiments are conducted in compute clusters with Python as the programming language (version 3.10), where we use 8x Nvidia H100 94GB NVL GPUs and 2x AMD EPYC 9554 CPU @ 3.1 GHz, 2x64 cores, and 24x 96GB RAM.

8 FORMAL LANGUAGES AND GRAMMARS.

In each experiment, we provide the LLM with strings sampled from a probabilistic formal language, with the learning task of generating unseen strings from the same language via syntactic pattern recognition. Underneath, a probabilistic formal language is represented by a *probabilistic formal grammars*, or simply *grammars* (Collins, 2013). Specifically, a grammar consists of two sets of symbols called the *non-terminals* and *terminals*, a set of rules to rewrite strings over these symbols that contain at least one nonterminal – also called the *production rules*, and a probability distribution over the production rules. Formally, a probabilistic formal grammar, is defined as a quintuple.

$$G = (\mathbf{N}, \mathbf{T}, \mathbf{R}, S, \mathbf{P})$$

where $\mathbf{N}$ is the set of non-terminals, $\mathbf{T}$ is the set of terminals (equivalently, tokens), $\mathbf{R}$ is the set of production rules, $S \in \mathbf{N}$ is the begin non-terminal, and $\mathbf{P}$ is the set of probabilities on production rules.

Formal languages are divided into well-known classes based on the *complexity* of the language membership problem, i.e., the *complexity* of the grammars needed to generate them (Chomsky, 1956). In this paper, we use one class of grammars, namely, hierarchical probabilistic context-free grammars (HPCFGs) (Allen-Zhu & Li, 2023). Specifically, our experiments are based on teaching LLMs languages represented by HPCFGs. We use HPCFGs because they are simple syntactically and can represent languages that are structurally similar to natural languages (Allen-Zhu & Li, 2023; Shi et al., 2022).

Description of Grammars and Identified Languages. In our experiments, we consider two generic structure for the considered grammars, one adapted from Allen-Zhu & Li (2023), namely G_1, G_2, G_5, G_7, G_9, and another is proposed by us, namely G_3, G_4, G_6, G_8.

In the first generic structure, such as G_1, the grammar has $\mathbf{N} = \{S, A7, A8, \ldots, A16\}$ and $\mathbf{T} = \{1, 2, 3, \ldots, 9\}$. The grammar has four levels of hierarchy: the non-terminals from top to bottom levels are $\{A16\}$, $\{A13, A14, A15\}$, $\{A10, A11, A12\}$, and $\{A7, A8, A9\}$, followed by terminals $\{1, 2, 3, \ldots, 9\}$. Each non-terminal (except the start non-terminal) has two expansion rules, consisting of non-terminals from the immediate lower level. Further, the expansion rules are probabilistic, where the sum of probabilities of all expansion rules from a given non-terminal is 1.

The second generic structure is inspired by bridging two HPCFGs together, and simulating a long range dependencies within the generated strings. Specifically, the sub-grammar at $B4$ and the sub-grammar at $E4$ are connected by non-terminal $C1_i$; and $E4$ ends with $T1_j$. Long range dependencies are communicated through $C1_i$ and $T1_j$, by enforcing $i = j$ at each expansion of $S5$.

In all cases, G_i produces a probabilistic context free language L_i. Figure 11 denotes the length distribution of different languages, and Figure 12 demonstrates how hierarchical non-terminals are applied in different positions in the representative strings.

Sampling Strings from a Formal Language. Given a language L generated by a HPCFG, we first need to obtain *training* samples, i.e., set of i.i.d. samples of strings from L. To *sample a string from the language*, we start from a special string in the grammar containing a single, distinguished nonterminal called the "start" or "root" symbol, and apply the production rules to rewrite the string repeatedly. If several rules can be used to rewrite the string at any stage, we sample one such rule from the probability distribution over the rules and apply it. We stop when we obtain a string containing terminal tokens only. This string is a sample drawn from the language. We can repeat this process to draw any number of i.i.d. samples from the language.

$S \rightarrow A16$ [1]
$A16 \rightarrow A15\ A14\ A13$ [0.50]
$A16 \rightarrow A13\ A15\ A14$ [0.50]
$A13 \rightarrow A11\ A12$ [0.50]
$A13 \rightarrow A12\ A11$ [0.50]
$A14 \rightarrow A11\ A10\ A12$ [0.50]
$A14 \rightarrow A10\ A11\ A12$ [0.50]
$A15 \rightarrow A12\ A11\ A10$ [0.50]
$A15 \rightarrow A11\ A12\ A10$ [0.50]
$A10 \rightarrow A7\ A9\ A8$ [0.50]
$A10 \rightarrow A9\ A8\ A7$ [0.50]
$A11 \rightarrow A8\ A7\ A9$ [0.50]
$A11 \rightarrow A7\ A8\ A9$ [0.50]
$A12 \rightarrow A8\ A9\ A7$ [0.50]
$A12 \rightarrow A9\ A7\ A8$ [0.50]
$A7 \rightarrow 3\ 1\ 2$ [0.50]
$A7 \rightarrow 1\ 2\ 3$ [0.50]
$A8 \rightarrow 6\ 5\ 4$ [0.50]
$A8 \rightarrow 6\ 4\ 5$ [0.50]
$A9 \rightarrow 9\ 8\ 7$ [0.50]
$A9 \rightarrow 8\ 7\ 9$ [0.50]

$S \rightarrow A16$ [1]
$A16 \rightarrow A15\ A14\ A13$ [0.95]
$A16 \rightarrow A13\ A15\ A14$ [0.05]
$A13 \rightarrow A11\ A12$ [0.95]
$A13 \rightarrow A12\ A11$ [0.05]
$A14 \rightarrow A11\ A10\ A12$ [0.95]
$A14 \rightarrow A10\ A11\ A12$ [0.05]
$A15 \rightarrow A12\ A11\ A10$ [0.95]
$A15 \rightarrow A11\ A12\ A10$ [0.05]
$A10 \rightarrow A7\ A9\ A8$ [0.95]
$A10 \rightarrow A9\ A8\ A7$ [0.05]
$A11 \rightarrow A8\ A7\ A9$ [0.95]
$A11 \rightarrow A7\ A8\ A9$ [0.05]
$A12 \rightarrow A8\ A9\ A7$ [0.95]
$A12 \rightarrow A9\ A7\ A8$ [0.05]
$A7 \rightarrow 3\ 1\ 2$ [0.95]
$A7 \rightarrow 1\ 2\ 3$ [0.05]
$A8 \rightarrow 6\ 5\ 4$ [0.95]
$A8 \rightarrow 6\ 4\ 5$ [0.05]
$A9 \rightarrow 9\ 8\ 7$ [0.95]
$A9 \rightarrow 8\ 7\ 9$ [0.05]

Figure 6: Production rules of G_1 (left) and G_2 (right). Compared to G_1, the grammar G_2 generates more skewed distribution (or lower entropy) strings, since one out of two production rules for each non-terminal is selected with higher probability.

In our experiments, we aim to split the sampled strings into training and test sets, which are disjoint but have a similar probability distribution over string occurrences. To realize this goal, we first sample a finite number of strings from the language. Then, we remove a random string from the finite set along with all its occurrences and put them into the training set. In the next iteration, we put the next random string with all its occurrences into the test set. The process repeats until the initial finite set is exhausted. Thus, we empirically try to obtain a similar probability distribution over string occurrences in the training and test sets.

$S \rightarrow S5$ [1]
$S5 \rightarrow B4\ C1_1\ E4\ T1_1$ [0.25]
$S5 \rightarrow B4\ C1_2\ E4\ T1_2$ [0.25]
$S5 \rightarrow B4\ C1_3\ E4\ T1_3$ [0.25]
$S5 \rightarrow B4\ C1_4\ E4\ T1_4$ [0.25]
$B4 \rightarrow B3$ [0.3333]
$B4 \rightarrow B3\ B3\ B3$ [0.3333]
$B4 \rightarrow B3\ B3$ [0.3333]
$B3 \rightarrow B2$ [0.3333]
$B3 \rightarrow B2$ [0.3333]
$B3 \rightarrow B2\ B2$ [0.3333]
$B2 \rightarrow B1$ [0.3333]
$B2 \rightarrow B1$ [0.3333]
$B2 \rightarrow B1\ B1\ B1$ [0.3333]
$B1 \rightarrow 2\ 9\ 3$ [0.3333]
$B1 \rightarrow 9\ 6\ 1$ [0.3333]
$B1 \rightarrow 1\ 8\ 6$ [0.3333]
$E4 \rightarrow E3$ [0.3333]
$E4 \rightarrow E3\ E3$ [0.3333]
$E4 \rightarrow E3\ E3\ E3$ [0.3333]
$E3 \rightarrow E2$ [0.3333]
$E3 \rightarrow E2\ E2$ [0.3333]
$E3 \rightarrow E2$ [0.3333]
$E2 \rightarrow E1\ E1$ [0.3333]
$E2 \rightarrow E1$ [0.3333]
$E2 \rightarrow E1\ E1\ E1$ [0.3333]
$E1 \rightarrow 5\ 6\ 5\ 9$ [0.3333]
$E1 \rightarrow 1\ 8\ 6\ 6$ [0.3333]
$E1 \rightarrow 1\ 5\ 1\ 5$ [0.3333]
$T1_1 \rightarrow 1$ [1]
$T1_2 \rightarrow 2$ [1]
$T1_3 \rightarrow 3$ [1]
$T1_4 \rightarrow 4$ [1]
$C1_1 \rightarrow 5$ [1]
$C1_2 \rightarrow 6$ [1]
$C1_3 \rightarrow 7$ [1]
$C1_4 \rightarrow 8$ [1]
$C1_5 \rightarrow 9$ [1]

$S \rightarrow S5$ [1]
$S5 \rightarrow B4\ C1_1\ E4\ T1_1$ [0.25]
$S5 \rightarrow B4\ C1_2\ E4\ T1_2$ [0.25]
$S5 \rightarrow B4\ C1_3\ E4\ T1_3$ [0.25]
$S5 \rightarrow B4\ C1_4\ E4\ T1_4$ [0.25]
$B4 \rightarrow B3$ [0.3333]
$B4 \rightarrow B3\ B3\ B3$ [0.3333]
$B4 \rightarrow B3\ B3$ [0.3333]
$B3 \rightarrow B2$ [0.3333]
$B3 \rightarrow B2$ [0.3333]
$B3 \rightarrow B2\ B2$ [0.3333]
$B2 \rightarrow B1$ [0.3333]
$B2 \rightarrow B1$ [0.3333]
$B2 \rightarrow B1\ B1\ B1$ [0.3333]
$B1 \rightarrow 2\ 9\ 3$ [0.95]
$B1 \rightarrow 9\ 6\ 1$ [0.025]
$B1 \rightarrow 1\ 8\ 6$ [0.025]
$E4 \rightarrow E3$ [0.3333]
$E4 \rightarrow E3\ E3$ [0.3333]
$E4 \rightarrow E3\ E3\ E3$ [0.3333]
$E3 \rightarrow E2$ [0.3333]
$E3 \rightarrow E2\ E2$ [0.3333]
$E3 \rightarrow E2$ [0.3333]
$E2 \rightarrow E1\ E1$ [0.3333]
$E2 \rightarrow E1$ [0.3333]
$E2 \rightarrow E1\ E1\ E1$ [0.3333]
$E1 \rightarrow 5\ 6\ 5\ 9$ [0.95]
$E1 \rightarrow 1\ 8\ 6\ 6$ [0.025]
$E1 \rightarrow 1\ 5\ 1\ 5$ [0.025]
$T1_1 \rightarrow 1$ [1]
$T1_2 \rightarrow 2$ [1]
$T1_3 \rightarrow 3$ [1]
$T1_4 \rightarrow 4$ [1]
$C1_1 \rightarrow 5$ [1]
$C1_2 \rightarrow 6$ [1]
$C1_3 \rightarrow 7$ [1]
$C1_4 \rightarrow 8$ [1]
$C1_5 \rightarrow 9$ [1]

Figure 7: Production rules of G_3 (left) and G_4 (right). Compared to G_3, the grammar G_4 generates more skewed distribution (or lower entropy) of strings, since one out of three production rules of non-terminal $B1$ and $E1$ is selected with higher probability.

$S \to A16$ [1]
$A16 \to A15\ A13$ [0.50]
$A16 \to A13\ A15\ A14$ [0.50]
$A13 \to A11\ A12$ [0.50]
$A13 \to A12\ A11$ [0.50]
$A14 \to A11\ A10\ A12$ [0.50]
$A14 \to A10\ A11\ A12$ [0.50]
$A15 \to A12\ A11\ A10$ [0.50]
$A15 \to A11\ A12\ A10$ [0.50]
$A10 \to A7\ A9\ A8$ [0.50]
$A10 \to A9\ A8\ A7$ [0.50]
$A11 \to A8\ A7\ A9$ [0.50]
$A11 \to A7\ A8\ A9$ [0.50]
$A12 \to A8\ A9\ A7$ [0.50]
$A12 \to A9\ A7\ A8$ [0.50]
$A7 \to 3\ 1$ [0.50]
$A7 \to 1\ 2\ 3$ [0.50]
$A8 \to 6\ 5$ [0.50]
$A8 \to 6\ 4\ 5$ [0.50]
$A9 \to 9\ 8\ 7$ [0.50]
$A9 \to 8\ 7$ [0.50]

$S \to S5$ [1]
$S5 \to B4\ C1_1\ E4\ T1_1$ [0.25]
$S5 \to B4\ C1_2\ E4\ T1_2$ [0.25]
$S5 \to B4\ C1_3\ E4\ T1_3$ [0.25]
$S5 \to B4\ C1_4\ E4\ T1_4$ [0.25]
$B4 \to B3$ [0.3333]
$B4 \to B3\ B3\ B3$ [0.3333]
$B4 \to B3\ B3$ [0.3333]
$B3 \to B2$ [0.3333]
$B3 \to B2$ [0.3333]
$B3 \to B2\ B2$ [0.3333]
$B2 \to B1$ [0.3333]
$B2 \to B1$ [0.3333]
$B2 \to B1\ B1\ B1$ [0.3333]
$B1 \to 2\ 9\ 3$ [0.3333]
$B1 \to 9\ 6\ 1$ [0.3333]
$B1 \to 1\ 8\ 6\ 2$ [0.3333]
$E4 \to E3$ [0.3333]
$E4 \to E3\ E3$ [0.3333]
$E4 \to E3\ E3\ E3$ [0.3333]
$E3 \to E2$ [0.3333]
$E3 \to E2\ E2$ [0.3333]
$E3 \to E2$ [0.3333]
$E2 \to E1\ E1$ [0.3333]
$E2 \to E1$ [0.3333]
$E2 \to E1\ E1\ E1$ [0.3333]
$E1 \to 5\ 6$ [0.3333]
$E1 \to 1\ 8\ 6\ 6$ [0.3333]
$E1 \to 1\ 5\ 1\ 5\ 5\ 9$ [0.3333]
$T1_1 \to 1$ [1]
$T1_2 \to 2$ [1]
$T1_3 \to 3$ [1]
$T1_4 \to 4$ [1]
$C1_1 \to 5$ [1]
$C1_2 \to 6$ [1]
$C1_3 \to 7$ [1]
$C1_4 \to 8$ [1]
$C1_5 \to 9$ [1]

Figure 8: Production rules of G_5 (left) and G_6 (right). These grammars are adapted from G_1 and G_3 respectively, by allowing non-uniform lengths of tokens in the lowest level production rules.

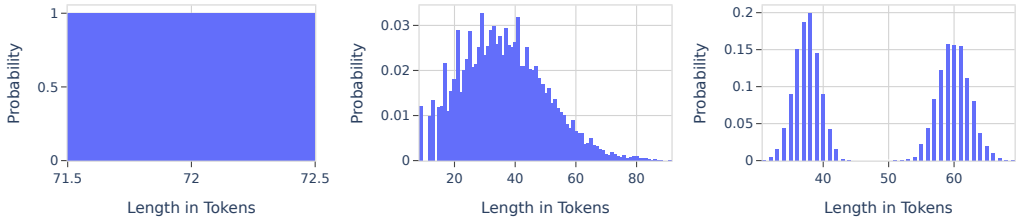
$S \to A16$ [1]
$A16 \to A15\ A13$ [0.50]
$A16 \to A13\ A15\ A14$ [0.50]
$A13 \to A11\ A12$ [0.50]
$A13 \to A12\ A11$ [0.50]
$A14 \to A11\ A10\ A12$ [0.50]
$A14 \to A10\ A11\ A12$ [0.50]
$A15 \to A12\ A11\ A10$ [0.50]
$A15 \to A11\ A12\ A10$ [0.50]
$A10 \to A7\ A9\ A8$ [0.50]
$A10 \to A9\ A8\ A7$ [0.50]
$A11 \to A8\ A7\ A9$ [0.50]
$A11 \to A7\ A8\ A9$ [0.50]
$A12 \to A8\ A9\ A7$ [0.50]
$A12 \to A9\ A7\ A8$ [0.50]
$A7 \to c\ a$ [0.50]
$A7 \to a\ b\ c$ [0.50]
$A8 \to f\ e$ [0.50]
$A8 \to f\ d\ e$ [0.50]
$A9 \to i\ h\ g$ [0.50]
$A9 \to h\ g$ [0.50]

$S \to S5$ [1]
$S5 \to B4\ C1_1\ E4\ T1_1$ [0.25]
$S5 \to B4\ C1_2\ E4\ T1_2$ [0.25]
$S5 \to B4\ C1_3\ E4\ T1_3$ [0.25]
$S5 \to B4\ C1_4\ E4\ T1_4$ [0.25]
$B4 \to B3$ [0.3333]
$B4 \to B3\ B3\ B3$ [0.3333]
$B4 \to B3\ B3$ [0.3333]
$B3 \to B2$ [0.3333]
$B3 \to B2$ [0.3333]
$B3 \to B2\ B2$ [0.3333]
$B2 \to B1$ [0.3333]
$B2 \to B1$ [0.3333]
$B2 \to B1\ B1\ B1$ [0.3333]
$B1 \to b\ i\ c$ [0.3333]
$B1 \to i\ f\ a$ [0.3333]
$B1 \to a\ h\ f\ b$ [0.3333]
$E4 \to E3$ [0.3333]
$E4 \to E3\ E3$ [0.3333]
$E4 \to E3\ E3\ E3$ [0.3333]
$E3 \to E2$ [0.3333]
$E3 \to E2\ E2$ [0.3333]
$E3 \to E2$ [0.3333]
$E2 \to E1\ E1$ [0.3333]
$E2 \to E1$ [0.3333]
$E2 \to E1\ E1\ E1$ [0.3333]
$E1 \to e\ f$ [0.3333]
$E1 \to a\ h\ f\ f$ [0.3333]
$E1 \to a\ e\ a\ e\ e\ i$ [0.3333]
$T1_1 \to a$ [1]
$T1_2 \to b$ [1]
$T1_3 \to c$ [1]
$T1_4 \to d$ [1]
$C1_1 \to e$ [1]
$C1_2 \to f$ [1]
$C1_3 \to g$ [1]
$C1_4 \to h$ [1]
$C1_5 \to i$ [1]

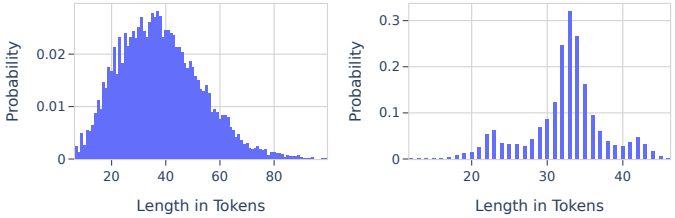
Figure 9: Production rules of G_7 (left) and G_8 (right). These grammars are adapted from G_5 and G_6 respectively, by replacing numerical tokens with Latin character tokens.

$S \to A13$ [1]
$A13 \to A10\ A12$ [0.8413447461]
$A13 \to A9\ A10$ [0.1586552539]
$A12 \to A5\ A7\ A5\ A8$ [0.8413447461]
$A12 \to A6\ A6$ [0.1586552539]
$A11 \to A5\ A7\ A8$ [0.8413447461]
$A11 \to A6$ [0.1586552539]
$A10 \to A6\ A7\ A5\ A7$ [0.8413447461]
$A10 \to A8\ A6$ [0.1586552539]
$A9 \to A6$ [0.8413447461]
$A9 \to A8\ A5\ A8$ [0.1586552539]
$A8 \to A1$ [0.8413447461]
$A8 \to A3$ [0.1586552539]
$A7 \to A1\ A2$ [0.8413447461]
$A7 \to A4\ A4$ [0.1586552539]
$A6 \to A2\ A1\ A3\ A4$ [0.8413447461]
$A6 \to A3\ A4\ A1$ [0.1586552539]
$A5 \to A1\ A4$ [0.8413447461]
$A5 \to A3$ [0.1586552539]
$A4 \to 4\ 3$ [0.8413447461]
$A4 \to 3\ 0\ 3$ [0.1586552539]
$A3 \to 2\ 4\ 1\ 0$ [0.8413447461]
$A3 \to 0\ 4\ 3$ [0.1586552539]
$A2 \to 0\ 4\ 3$ [0.8413447461]
$A2 \to 4\ 3\ 0\ 2$ [0.1586552539]
$A1 \to 1$ [0.8413447461]
$A1 \to 2$ [0.1586552539]

Figure 10: Production rules of G_9. The hierarchical grammar has a maximum depth 4, maximum breadth 4, rules per non-terminals 2, and 5 numerical tokens.

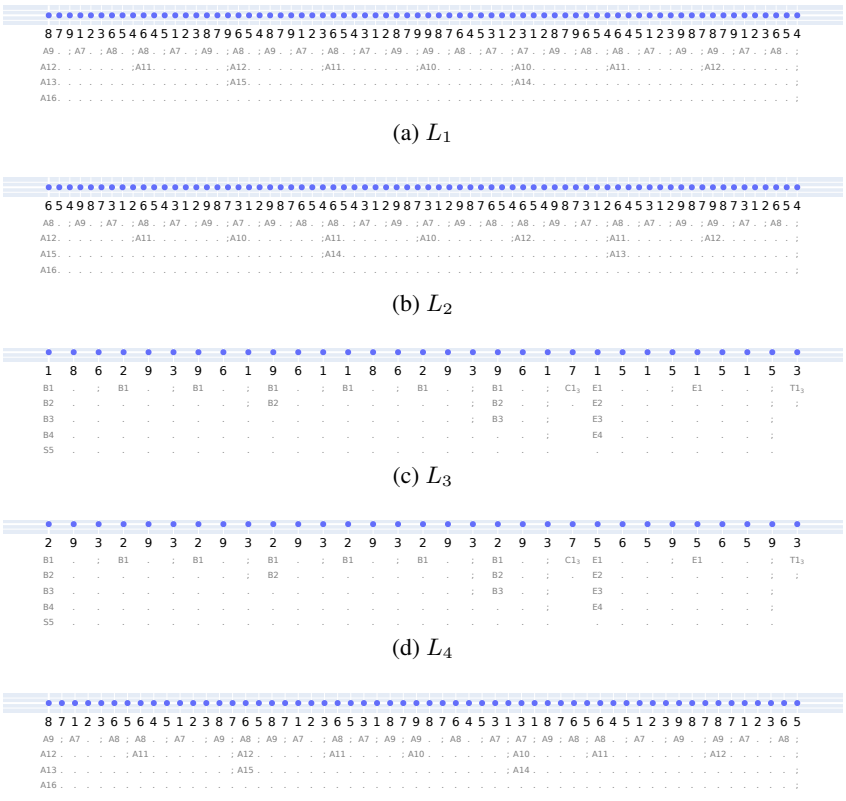


(a) L_1 (also L_2) (b) L_3 (also L_4) (c) L_5 (also L_7)



(d) L_6 (also L_8) (e) L_9

Figure 11: Length distribution of strings sampled from multiple formal languages. The length distribution is based on 10000 sampled strings per language.



(a) L_1

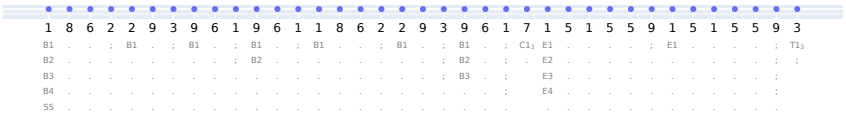
(b) L_2

(c) L_3

(d) L_4

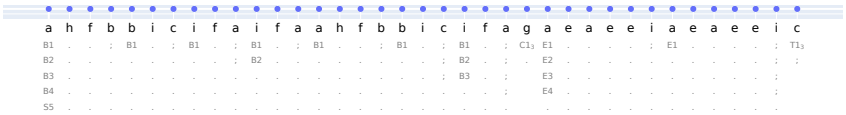
(e) L_5

Figure 12: Representative strings from different languages, annotated with non-terminals applied in different positions by the respective hierarchical grammar.

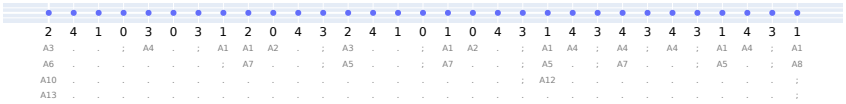


(a) L_6

(b) L_7



(c) L_8



(d) L_9

Figure 13: Representative strings from different languages, annotated with non-terminals applied in different positions by the respective hierarchical grammar.