Proceedings of the 7th Symposium on Advances in Approximate Bayesian Inference, 2025 1–17

Transcending Bayesian Inference: Transformers Extrapolate Rules Compositionally under Model Misspecification

Anonymous Authors
Anonymous Institution

Abstract

LLMs' intelligent behaviour, such as emergent reasoning and in-context learning abilities have been interpreted as implicit Bayesian inference (IBI). IBI considers the training data as a mixture, infers the underlying latent parameters thereof, and makes predictions on the training data consistent with explicit Bayesian inference. When the test prompts are out-of-distribution, Bayesian inference over the training mixture components becomes suboptimal due to model misspecification. We pre-train Transformer models for implicit Bayesian inference, and investigate whether they can transcend this behaviour under model misspecification. Our experiments demonstrate that Transformers generalize compositionally, even when the Bayesian posterior is undefined. We hypothesize this behavior is due to Transformers learning general algorithms instead of only fitting the training mixture.

1. Introduction

Autoregressive language models (AR LMs) trained on next-token prediction show remarkable emergent abilities, such as reasoning (Ouyang et al., 2022; Wei et al., 2022; Touvron et al., 2023), and in-context learning (ICL) (Xie et al., 2022; Min et al., 2022; Zhang et al., 2023). A common narrative for explaining these behaviors, and model intelligence in general, is implicit Bayesian inference (IBI), which posits that models trained on mixture data infer concepts form in-distribution prompts, and produce completions consistent with the Bayesian posterior predictive. Such interpretations were provided for ICL (Xie et al., 2022; Wang et al., 2024) and more general language model (LM) inference (Dalal and Misra, 2024).

This view's advantage is that Bayesian inference can be loosely seen as a form of intelligence: the posterior predictive is constructed through systematic updates, and converges to the true data distribution for well-specified Bayesian models (Van der Vaart, 1998).

However, it is well known that Bayesian models struggle to make accurate predictions whenever the model is misspecified, i.e. the true parameter/data-distribution has zero weight under the prior (Masegosa, 2020; Morningstar et al., 2022). Although large language models are trained on massive mixture distributions exposing the model to a high variety of tasks (e.g., books, medical texts, journalism, technical manuals), it is unclear whether each possible user prompt is sufficiently close to a type of task seen in the training data. Thus, LMs might still operate under model misspecification. In that case, their success is even more remarkable, and its understanding is even less clear. To advance our understanding, we attempt to answer the question *can models pre-trained for implicit Bayesian inference (IBI) transcend this behaviour when prompted out-of-distribution (OOD)?*

We train decoder-only Transformers (Vaswani et al., 2017) on simple synthetic data mixtures, such as coinflips and formal language sequences, encouraging IBI in the training. We create compositional out-of-distribution (OOD) tasks controlling the level of model

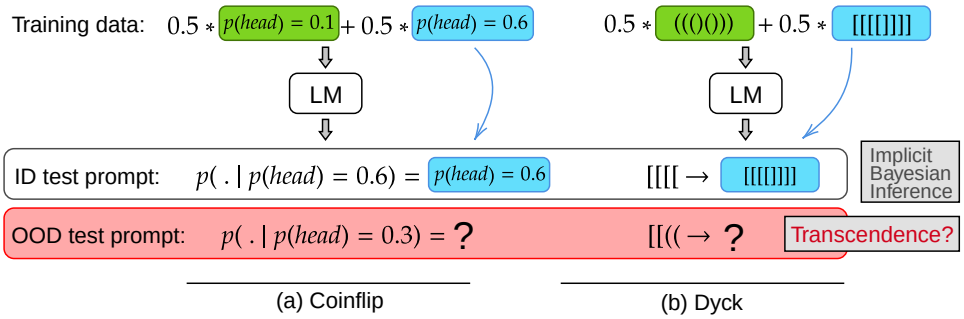


Figure 1: **Our experimental setting:** we train Transformer models on probabilistic mixtures for implicit Bayesian inference, and evaluate whether they can transcend this behaviour on OOD prompts. For details, refer to §§ 4 and 5

misspecification (see Fig. 1). On these OOD prompts, model behavior is weakly constrained by the training data, hence other kinds of behaviours are possible. We find that models can extrapolate much better than implied by IBI, and can even generalise meaningfully where the Bayesian posterior is undefined due to zero-likelihood under all components—potentially suggesting some implicit inductive bias in the model. This demonstrates that *Transformers can display intelligent behaviour transcending IBI over the training data in OOD tasks.*

Our work was inspired by Zhang et al. (2024), who introduced the term *transcendence* for the event when language models trained on chess scripts produced by separate experts beat all experts' performance. They assess performance on the same task (playing chess), whereas we consider completely separate OOD test tasks. There are several other important lines of research in the literature, which, due to limitations of space, we detail in Appx. A.

2. Implicit Bayesian Inference in Language Models

We study next-token prediction in autoregressive probabilistic models. Let $x_{1:L}$ denote a token sequence from the set of all possible token sequences $\mathcal{X}$ of fixed maximal length L. An autoregressive probabilistic model is defined a collection $\{p(x_i|x_{1:i-1},\theta); 1 \leq i \leq L\}$ of conditional distributions with parameters θ in some fixed parameter space Θ.

To define implicit Bayesian inference, we assume that our data can be viewed as a mixture of simpler components, i.e. $p(x_1,\ldots,x_L) = \int_\Theta p(\theta)p(x_1,\ldots,x_L \mid \theta)d\theta$. A Bayesian model perfectly fitting this data produces its next-token predictions according to the posterior predictive $p(x_i \mid x_{i-1},...,x_1) = \int_\Theta p(x_i \mid \theta, x_{i-1},...,x_1)p(\theta \mid x_{i-1},...,x_1)d(\theta)$, which marginalizes the likelihood of x_i given θ and the prompt with respect to the posterior of θ. This can be interpreted as predicting the next-token using the latent concepts that best fit the prompt.

Accordingly, we say that a model performs IBI, if after training on such a mixture data distribution, its produced next-token predictions align with the next-token probabilities of the Bayesian posterior predictive (over the train distribution). The word *implicit* means that the model does not need to explicitly represent the latent variables or compute prior and posterior probabilities.

Typically, model training can be interpreted as meta-learning IBI. Specifically, training autoregressive models on the next-token prediction objective via minimizing cross-entropy encourages the model to perfectly fit the training mixture (provided its expressivity). This

yields provable implicit Bayesian inference, as perfectly learning the mixture components means we can use them in Bayes' rule. In practice, models don't exactly reach the minimum of the expected loss, and thus only approximate an explicit Bayesian model.

IBI is desirable, as it is asymptotically optimal on in-distribution data, i.e., it generalizes statistically. This only holds for well-specified Bayesian models, i.e., models where the true parameter θ is contained in Θ (Van der Vaart, 1998). However, when evaluated on OOD prompts from components outside of Θ, the Bayesian posterior predictive is no longer asymptotically optimal, leading IBI to also perform sub-optimally in such settings.

3. Formal Languages

Formal languages are abstractions of natural languages, consisting of sequences of symbols from a fixed alphabet that obey a specific set of rules called formal grammar. Although they are much simpler than natural languages, their clear rules allow for systematically studying OOD generalization (Delétang et al. (2024), Mészáros et al. (2024)), and they enjoy real-world applications for code generation and mathematical reasoning in LLMs. Dyck languages consist of opening and closing brackets, and the rules depend on the depth (number of bracket types) and whether the language is nested. We call a Dyck language *nested* if, in addition to all bracket types being correctly matched (e.g. `([])`), they are also correctly nested, just like in arithmetic expressions (e.g. `[()]`) and *not nested (NN)* otherwise. When referring to a Dyck$-i$ language, we mean the nested version, unless explicitly specified as NN Dyck$-i$. All Dyck languages we consider together with their rules can be found in are in Tab. 1 in Appx. A. According to the categorization of Chomsky (1956), nested Dyck languages are context-free, that is, generating tokens obeys the same rule, irrespective of the other tokens in the sequence. Our work uses Dyck languages to test whether models trained on mixtures of simpler Dyck languages can extrapolate on more complex versions.

4. Coinflip Experiments

Datasets. We train on a mixture of two i.i.d. Bernoulli components with success probabilities 0.1 and 0.6, respectively. For each data point, we select the mixture component with equal probability. Our test prompts have length 100, making draws from most test distributions very unlikely under the train data. The test prompts are drawn i.i.d. from:

1. a single Bernoulli with success probability $p \in [0.1, 0.6]$ (Fig. 2)
2. a single Bernoulli with success probability $p \in [0, 0.1) \cup (0.6, 1.0]$ (Fig. 2)
3. a two-state Markov chain with stationary distribution $[0.4, 0.6]$, matching the portion of successes in the second component of the train data, a Bernoulli with parameter 0.6. We initialise the Markov chain at its stationary distribution. (Fig. 3)
4. a single Bernoulli with success probability 0.1, prepended with an unseen token (Fig. 4)

The above tasks are in order of increasing difficulty: models can interpolate between the train components in the first task, but not in the second. The third task breaks the independence structure seen in the train data, while the fourth contains prompts that have zero-probability and thus are completely OOD.

Results. Our results show that Transformers accurately learned the correct next-token probabilities of p(head) and p(tail) on the train data components, i.e., when $p(\text{head}) = 0.1$

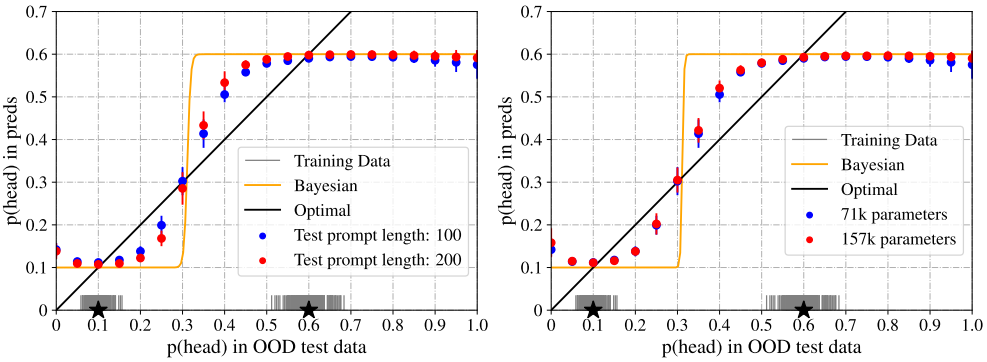


Figure 2: **Predicted p(head) next-token probabilities for OOD coinflip data with different levels of difficulty**: The training data was a mixture of two components with 0.1 and 0.6 head probabilities. The optimal solution is shown as the $y = x$ **black** line, the Bayesian estimate (switching between the two components at $(0.1 + 0.6)/2 = 0.35$ as , the two components' probabilities with a star symbol (the training data samples empirical probabilities are plot as discrete lines). **(Left:)** the negligible effect of test prompts length on OOD performance; **(Right:)** the negligible effect of model size. The figures show that Transformers transcend IBI when p(head) $\in [0.1, 0.6]$, whereas they collapse to one components (thus, do IBI) outside. Further ablations are included in Fig. 6

or $= 0.6$ in the test prompt, the model predicted accordingly (Fig. 2). Therefore, the Transformers closely approximate implicit Bayesian inference on the ID prompts.

Low probability i.i.d. coinflips (1, 2). The performance of Transformers on low-probability OOD coinflips varies based on whether the OOD success probability p(head) is within or outside of the interval $[0.1, 0.6]$, i.e., the set of convex combinations of the success probabilities in the train data. Transformers can extrapolate better than explicit Bayesian inference within the convex combination, but their performance falls below Bayesian inference outside. We hypothesize that tasks within the convex combination are easier in terms of compositionality: the correct p(head) probabilities can be obtained by weighted majority voting of the train components. Increasing the size of the Transformer and the test prompt by a factor of 2 length resulted in negligible changes in performance, but increasing these parameters further decreased extrapolation performance (see Fig. 6 in Appx. C).

Markov chain (3). Although the prompts in the train data are all i.i.d., the Transformer's estimated transition matrix is slightly closer to the true transitions and negative log likelihood than those of a Bayesian model Fig. 3. This indicates that the Transformer captured in its predictions a limited amount of the autocorrelations seen only in the test prompts.

Zero-probability prompts (4). Transformers were able to complete the zero-probability prompts which we constructed by prepending Bernoulli(0.1) draws with an unseen token. However, we observed a small but consistent upwards bias in the p(head) probabilities. The upwards bias may be due to the fact that the unseen token (5) had higher value than the tokens representing tails and heads (3 and 4, respectively), indicating that the model may have computed token sums in the prompts.

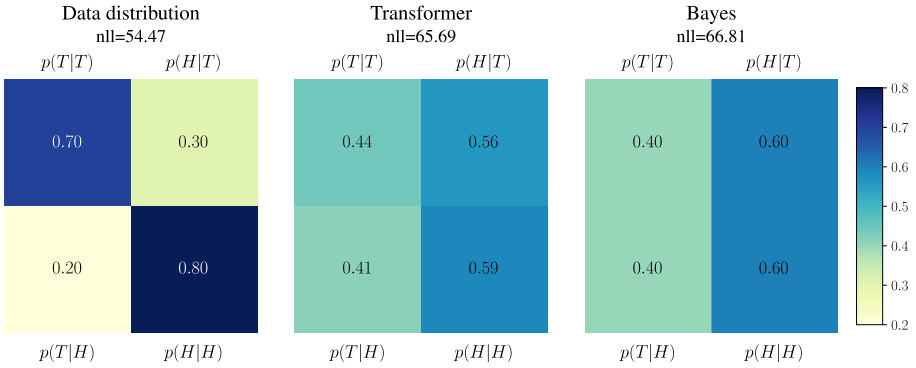
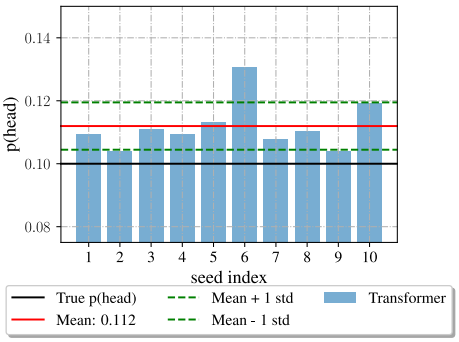


Figure 3: **Transition probabilities and negative log likelihood on OOD Markov chain data:** Models were evaluated on prompts drawn from a Markov chain with transition probabilities given in the leftmost figure, initialised at its stationary distribution p(head)= 0.6. The Transformers' transition probabilities were estimated from the next-token probabilities based on the previous token. Compared to the Bayes model, where each token is independent, the Transformer captured a limited amount of the autocorrelations in the test data.

Figure 4: **Predicted p(head) next-token probabilities on zero-probability prompts for ten seeds in the coinflip setting:** The model was trained on a mixture of two Bernoulli distributions with head probabilities 0.1, and 0.6, and tested on prompts drawn from the first (i.e., with p(head)=0.1) component that were made OOD by prepending them with an unseen token. The unseen token introduces a small upwards bias into the Transformer's p(head) probabilities, but their predictions are still close to the true value of p(head) in the data.



5. Formal Language Experiments

Datasets. We define more zero-probability test settings using mixtures of Dyck languages. The training data comes from mixtures of Dyck$-i$, for $i \in \{1, 2\}$ languages, containing either two or three components, depending on the task—the difference between these components is the type of parenthesis we use. Each component shares the same depth i. The test prompts are from a Dyck$-\{i+1\}$ or NN (not nested) Dyck$-\{i+1\}$. All experimental scenarios are detailed in Tab. 2 in Appx. A. Despite all involving zero-probability test prompts, the tasks differ in terms of the degree of extra information the model needs to deduce from the prompt to complete it according to the test languages' rules.

Results. Transformers are often able to perfectly learn the training data mixtures, as indicated by Fig. 5. When evaluating in the OOD settings, we compared the models' performance to chance-level accuracies, which were estimated a upper bounds on the chance of completing the test prompts by sampling each token (excluding the padding token) with equal probability. Interestingly, Transformers can massively beat chance level accuracies when trained on mixtures of Dyck-2 languages, but not when they are trained on Dyck-1

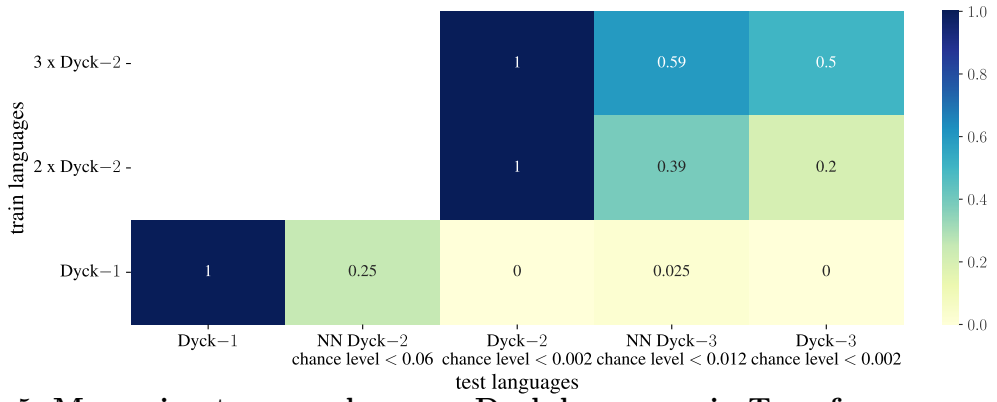


Figure 5: **Measuring transcendence on Dyck languages in Transformers, result of the best model (out of 10 seeds):** We train Transformers on simpler Dyck languages ($y-$axis), then test whether the models can finish test prompts for more complex (OOD) languages. Including more structure in the training language components (i.e., generalizing from Dyck-2 to Dyck-3) is easier for the Transformer than generalizing from simpler Dyck-1 language components. NN refers to *not nested*

components. We hypothesize that this is due to the models having seen the idea of nesting in the Dyck-2 components, which transfers in part to Dyck-3. In addition, seeing all three versions of Dyck-2 languages helps a lot in generalising nesting to depth three. We also observed strong variance over our 10 seeds. Hence, we supply additional plots in Appx. C showcasing the average performance of models in each task, as well as the standard deviations.

6. What is required for extrapolation?

Our tasks were constructed so that IBI learns each component as separate information, but the tasks can also be solved by representing the train components in a common, more general framework. For example, Bernoulli(p) coinflips can be accurately completed by counting the fraction of heads in the prompt, and Dyck-2 languages can be seen as subcases of Dyck-3 languages covering different parts of its structure. We hypothesize that transcendence over IBI is an indication of (some) learning of a general algorithm, and that compositionality in the train data helps models find generalist solutions. A similar explanation is often provided for grokking (Power et al., 2022), where models that perfectly generalise on algorithmic data—despite trained with only with limited examples—tend to capture the inherent structural symmetries present in the test data. This suggests the development of a generalist model that effectively internalizes the underlying rules. However, our work goes beyond in hypothesizing that similar behaviour is possible when training on a mixture of compositionally related distributions instead of algorithmically related single datapoints.

7. Conclusion

We have empirically shown that when trained on data mixtures, Transformers often extrapolate intelligently under model misspecification. Even though their training encourages implicit Bayesian inference, they can transcend this behaviour, most markedly in compositional tasks. We hypothesize that Transformers might learn generalist learning algorithms, which model the training data as special cases within a more general framework.

References

Kartik Ahuja and Amin Mansouri. On Provable Length and Compositional Generalization, February 2024. URL http://arxiv.org/abs/2402.04875. arXiv:2402.04875 [cs, stat].

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end Algorithm Synthesis with Recurrent Networks: Logical Extrapolation Without Overthinking, October 2022. URL http://arxiv.org/abs/2202.05826. arXiv:2202.05826 [cs].

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers to recognize formal languages, 2020. URL https://arxiv.org/abs/2009.11264.

Jack Brady, Roland S. Zimmermann, Yash Sharma, Bernhard Schölkopf, Julius von Kügelgen, and Wieland Brendel. Provably Learning Object-Centric Representations, May 2023. URL http://arxiv.org/abs/2305.14229. arXiv:2305.14229 [cs].

Jack Brady, Julius von Kügelgen, Sébastien Lachapelle, Simon Buchholz, Thomas Kipf, and Wieland Brendel. Interaction Asymmetry: A General Principle for Learning Composable Abstractions, November 2024. URL http://arxiv.org/abs/2411.07784. arXiv:2411.07784 [cs].

N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956. doi: 10.1109/TIT.1956.1056813.

Siddhartha Dalal and Vishal Misra. Beyond the black box: A statistical model for llm reasoning and inference, 2024. URL https://arxiv.org/abs/2402.03175.

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. Neural networks and the chomsky hierarchy, 2023.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language Modeling Is Compression, March 2024. URL http://arxiv.org/abs/2309.10668. arXiv:2309.10668 [cs, math].

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality, 2023.

Javid Ebrahimi, Dhruv Gelda, and Wei Zhang. How can self-attention networks recognize dyck-n languages?, 2020. URL https://arxiv.org/abs/2010.04303.

OpenAI et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Fabian Falck, Ziyu Wang, and Chris Holmes. Is in-context learning in large language models bayesian? a martingale perspective. ICML'24. JMLR.org, 2024.

PyTorch Lightning Falcon. Pytorch lightning: Accelerated research from prototypes to production. https://github.com/PyTorchLightning/pytorch-lightning, 2019.

Sungjun Han and Sebastian Padó. Towards Understanding the Relationship between In-context Learning and Compositional Generalization, March 2024. URL http://arxiv.org/abs/2403.11834. arXiv:2403.11834 [cs].

Sébastien Lachapelle, Divyat Mahajan, Ioannis Mitliagkas, and Simon Lacoste-Julien. Additive Decoders for Latent Variables Identification and Cartesian-Product Extrapolation, July 2023. URL http://arxiv.org/abs/2307.02598. arXiv:2307.02598 [cs, stat].

Brenden M. Lake and Marco Baroni. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121, November 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06668-3. URL https://www.nature.com/articles/s41586-023-06668-3. Publisher: Nature Publishing Group.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. URL http://arxiv.org/abs/1711.05101. arXiv:1711.05101 [cs, math].

Andres R. Masegosa. Learning under model misspecification: Applications to variational and ensemble methods, 2020. URL https://arxiv.org/abs/1912.08335.

R. Thomas McCoy, Shunyu Yao, Dan Friedman, Mathew D. Hardy, and Thomas L. Griffiths. Embers of autoregression show how large language models are shaped by the problem they are trained to solve. *Proceedings of the National Academy of Sciences*, 121(41): e2322420121, October 2024. doi: 10.1073/pnas.2322420121. URL https://www.pnas.org/doi/10.1073/pnas.2322420121. Publisher: Proceedings of the National Academy of Sciences.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work?, 2022.

Pablo Moreno-Muñoz, Pol G. Recasens, and Søren Hauberg. On masked pre-training and the marginal likelihood, 2023. URL https://arxiv.org/abs/2306.00520.

Warren R. Morningstar, Alex Alemi, and Joshua V. Dillon. Pacm-bayes: Narrowing the empirical risk gap in the misspecified bayesian regime. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 8270–8298. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/morningstar22a.html.

Anna Mészáros, Szilvia Ujváry, Wieland Brendel, Patrik Reizinger, and Ferenc Huszár. Rule Extrapolation in Language Modeling: A Study of Compositional Generalization on OOD Prompts. November 2024. URL https://openreview.net/forum?id=Li2rpRZWjy.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of transformers with simple arithmetic tasks, 2021.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Steven Pinker. *The language instinct*. William Morrow, New York, NY, December 1994.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets, January 2022. URL http://arxiv.org/abs/2201.02177. arXiv:2201.02177 [cs].

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. page 24, 2018.

Rahul Ramesh, Ekdeep Singh Lubana, Mikail Khona, Robert P. Dick, and Hidenori Tanaka. Compositional Capabilities of Autoregressive Transformers: A Study on Synthetic, Interpretable Tasks, February 2024. URL http://arxiv.org/abs/2311.12997. arXiv:2311.12997 [cs].

Patrik Reizinger, Szilvia Ujváry, Anna Mészáros, Anna Kerekes, Wieland Brendel, and Ferenc Huszár. Position: Understanding LLMs Requires More Than Statistical Generalization. June 2024. URL https://openreview.net/forum?id=pVyOchWUBa.

Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. Randomized Positional Encodings Boost Length Generalization of Transformers, May 2023. URL http://arxiv.org/abs/2305.16843. arXiv:2305.16843 [cs, stat].

Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Seyed Mehran Kazemi, Najoung Kim, and He He. Testing the general deductive reasoning capacity of large language models using ood examples, 2023.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. LSTM networks can perform dynamic counting. In Jason Eisner, Matthias Gallé, Jeffrey Heinz, Ariadna Quattoni, and Guillaume Rabusseau, editors, *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3905. URL https://aclanthology.org/W19-3905/.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

A. W. Van der Vaart. "10.2 bernstein–von mises theorem", asymptotic statistics. *Cambridge University Press*, 1998.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning, 2024. URL https://arxiv.org/abs/2301.11916.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Gail Weiss, Yoav Goldberg, and Eran Yahav. On the practical computational power of finite precision rnns for language recognition, 2018. URL https://arxiv.org/abs/1805.04908.

Thaddäus Wiedemer, Jack Brady, Alexander Panfilov, Attila Juhos, Matthias Bethge, and Wieland Brendel. Provable Compositional Generalization for Object-Centric Learning, October 2023a. URL http://arxiv.org/abs/2310.05327. arXiv:2310.05327 [cs].

Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. Compositional Generalization from First Principles, July 2023b. URL http://arxiv.org/abs/2307.05596. arXiv:2307.05596 [cs, stat].

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference, 2022.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On Layer Normalization in the Transformer Architecture, June 2020. URL http://arxiv.org/abs/2002.04745. arXiv:2002.04745 [cs, stat].

Edwin Zhang, Vincent Zhu, Naomi Saphra, Anat Kleiman, Benjamin L. Edelman, Milind Tambe, Sham M. Kakade, and Eran Malach. Transcendence: Generative Models Can Outperform The Experts That Train Them, October 2024. URL http://arxiv.org/abs/2406.11741. arXiv:2406.11741.

Ruiqi Zhang, Spencer Frei, and Peter L. Bartlett. Trained Transformers Learn Linear Models In-Context, October 2023. URL http://arxiv.org/abs/2306.09927. arXiv:2306.09927 [cs, stat].

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization, 2023. URL https://arxiv.org/abs/2310.16028.

Appendix A. Related Work

Implicit Bayesian Inference in Language Models. Several works interpret LLM capabilities through a Bayesian lens, often as implicit Bayesian inference (Moreno-Muñoz et al., 2023). Xie et al. (2022) show in a theoretical setting that, on a mixture of HMMs data, IBI over the components achieves in-context learning. Wang et al. (2024) give a similar interpretation for ICL, calling LLMs implicit topic models and providing evidence for real-world LLMs. However, there are works critiquing the validity of the Bayesian connection: Falck et al. (2024) demonstrates that ICL fails to display the necessary symmetry properties of Bayesian models. Dalal and Misra (2024) study LLM inference in general, positing that next-token prediction is consistent with Bayesian updating of next-token multinomial probabilities for each possible prompt. Our focus differs as we are not assessing the technical validity of the IBI interpretation. We train Transformers to approximate IBI on the training data and ask what the consequences are in OOD scenarios. There are several extensions of the Bayesian framework attempting to address model misspecification (Masegosa, 2020; Morningstar et al., 2022), proposing normative models of LLM inference that may explain OOD behavior. The scope of this work is only to demonstrate the incompleteness of the IBI narrative by showing examples where intelligent OOD extrapolation is not due to IBI.

Formal languages. Formal languages offer a controlled framework for modelling complex natural languages, enabling the analysis of language model behaviour on fundamental tasks, such as counting and learning hierarchical structures in the context of Dyck languages. Mészáros et al. (2024) demonstrated that common machine learning architectures can accurately complete in-distribution prompts for $a^n b^n$-type counting tasks, Dyck-n languages, and excluding the Transformer, tasks involving parity calculation. When testing OOD, most works focus on length generalization Delétang et al. (2023), finding that LSTMs can generalize well to longer sequences on $a^n b^n$-type grammars (Weiss et al., 2018) and on the Dyck-1 grammar (Suzgun et al., 2019; Weiss et al., 2018).Transformers only perform well on Dyck-n languages when a start-of-sequence token was included Ebrahimi et al. (2020), and fail to length generalize on parity calculation tasks (Bhattamishra et al., 2020; Zhou et al., 2023). Our work on Dyck languages is most similar Mészáros et al. (2024) studying rule extrapolation, the ability to complete prompts with partly broken rules when trained on formal languages. Transformers excel on context-free and context-sensitive languages including Dyck-2 languages. Our tasks differ in that we train models on mixture data with components obeying separate rules, and evaluate them on their composition. This makes our paper a contribution towards the compositional generalization research, too.

Compositional generalization. Most works on compositional generalization focus on computer vision (Wiedemer et al., 2023a,b; Brady et al., 2024, 2023; Lachapelle et al., 2023). For natural language, recent works started to studying OOD generalization, but mostly focusing on length generalization (Ahuja and Mansouri, 2024; Han and Padó, 2024; Ramesh et al., 2024; Lake and Baroni, 2023; Nogueira et al., 2021; Dziri et al., 2023; Saparov et al., 2023), showing that the positional encoding has a significant role in length generalization (Ruoss et al., 2023; Bansal et al., 2022). For some cases, theoretical results also exist (Ahuja and Mansouri, 2024). The relevance of studying compositional generalization in language models stems from the compositional nature of natural (and formal) languages (Pinker, 1994). Studying OOD generalization in pre-trained models is non-trivial as spurious correlations or the dominance of particular settings in the training data can lead to substantial performance differences, even in the same task (McCoy et al., 2024). Hence we follow previous works and train our models from scratch (Reizinger et al., 2024; Mészáros et al., 2024).

Appendix B. Experimental details

Reproducibility and codebase. We use PyTorch (Paszke et al., 2019), PyTorch Lightning (Falcon, 2019), and HuggingFace Transformers (Wolf et al., 2020). If accepted, we plan to make our codebase and experimental logs publicly available.

Coinflip data. The training data is generated by sampling a binary component variable with success probability 0.5, and based on the outcome, drawing i.i.d. Bernoullis with success probabilities 0.1 and 0.6, respectively. The coinflip outcomes are represented with tokens 3 and 4. Train prompts have length 256. The test prompts have length 100, and are either i.i.d. Bernoullis, i.i.d. Bernoullis prepended with an unseen token (5), or independent draws from a binary Markov chain.

Dyck language data. For training, we generate data from mixtures of Dyck$-i$, for $i \in \{1, 2\}$ languages, containing either two or three components, depending on the task. Each component shares the same depth i. Train prompts have maximal length 256. Besides the parentheses-type tokens `(` (3), `)` (4), `[` (5), `]` (6), `{` (7), `}` (8), we use SOS (0), EOS (1), and PAD (2) tokens. Our test prompts are from a Dyck$-i$ or NN Dyck$-i$ grammar with depth i one higher than the train data's components. They have length 10, including SOS as the first token, and start with 3 opening tokens containing all possible opening tokens in the test language, in arbitrary order, followed by 6 tokens forming a correct sequence from the test language. For example, if the test language is Dyck-3, an example test prompt is `( { [ { ( { } ) }`.

Models. We use a Transformer decoder (Vaswani et al., 2017) in flavor of the decoder-only GPT models (Radford and Narasimhan, 2018; Radford et al., 2018; et al., 2024). We apply standard positional encoding, layer normalization, ReLU activations, the AdamW optimizer (Loshchilov and Hutter, 2019) with inverse square root learning rate schedule (Xiong et al., 2020). The models are initialized with the full token dictionary, including tokens potentially not present in the train data distribution. For the coinflip experiments, next-tokens were selected using sampling decoding, while we used greedy decoding for the Dyck experiments, consistent with Mészáros et al. (2024). In the coinflip and Dyck experiments,

the model can predict up to length 250 and 300, respectively, and is trained for 4000 and $50,000$ epochs, respectively, using the standard cross entropy (CE) loss.

Metrics. In the coinflip experiments, we measure the probability of predicting heads or tails as the next token. In the Dyck experiments, we evaluate grammatical accuracies in both the train and test grammars, assessing the full, completed prompt. That is, we check whether all grammar rules for the train/test languages. Further details on our experimental setup are provided in Appx. A.

Language	**Vocabulary**	**Rules**	**Classification**
Dyck−1	(	(R1) (and) matched	context-free
Dyck−2	(, [	(R1) (and) matched and nested (R2)[and] matched and nested	context-free
NN Dyck−2	(, [	(R1) (and) matched (R2)[and] matched	context-sensitive
Dyck−3	(, [, {	(R1) (and) matched and nested (R2)[and] matched and nested (R3){ and } matched and nested	context-free
NN Dyck−3	(, [, {	(R1) (and) matched (R2)[and] matched (R3){ and } matched	context-sensitive

Table 1: **All Dyck languages studied in our paper:** We include the types of brackets used in the language in the Vocabulary columns (closing versions are also included), we list their rules and their classification under the Chomsky hierarchy (Chomsky, 1956)

Train language components			**Test language**
Dyck−2 (, [	Dyck−2 (, {	Dyck−2 [, {	NN Dyck−3(, [, {
Dyck−2 (, [	Dyck−2 (, {		NN Dyck−3 (, [, {
Dyck−2 (, [	Dyck−2 (, {	Dyck−2 [, {	Dyck−3 (, [, {
Dyck−2 (, [	Dyck−2 (, {		Dyck−3 (, [, {
Dyck−1 (	Dyck−1 [		NN Dyck−2 (, [
Dyck−1 (	Dyck−1 [	Dyck−1 {	NN Dyck−3 (, [{
Dyck−1 (	Dyck−1 [		Dyck−2 (, [
Dyck−1 (	Dyck−1 [	Dyck−1 {	Dyck−3 (, [{

Table 2: **Dyck languages used for training and testing:** We list the scenarios in increasing order of task difficulty. Please refer to Tab. 1 for the description of the language rules

Table 3: Transformer parameters for coinflip experiments

Parameter	Value (normal)
Model	Transformer decoder
Number of layers	4
Dropout probability	0.1
Model dimension	8
Feedforward dimension	1024
Number of attention heads	4
Layer norm ϵ	6e−3
Activation	ReLU
Optimizer	AdamW
Learning rate scheduler	inverse square root
Batch size	128
Learning rate	2e−3
Prompt prediction cutoff length	256
Number of epochs	4000

Table 4: Transformer parameters for Dyck experiments

Parameter	Value (normal)
Model	Transformer decoder
Number of layers	7
Dropout probability	0.1
Model dimension	10
Feedforward dimension	1024
Number of attention heads	5
Layer norm ϵ	6e−3
Activation	ReLU
Optimizer	AdamW
Learning rate scheduler	inverse square root
Batch size	128
Learning rate	2e−3
Prompt prediction cutoff length	300
Number of epochs	50,000

Appendix C. Further experimental results

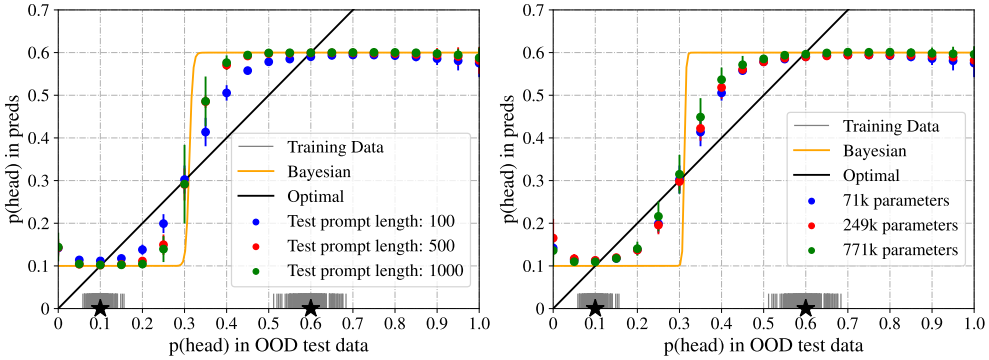


Figure 6: **Predicted p(head) next-token probabilities for OOD coinflip data with different levels of difficulty**: The training data was a mixture of two components with 0.1 and 0.6 head probabilities. The optimal (ground-truth) solution is shown as the $y = x$ **black** line, the Bayesian estimate (switching between the two components at $(0.1 + 0.6)/2 = 0.35$ as , the two components' probabilities with a star symbol (the training data samples empirical probabilities are plot as discrete lines). **(Left:)** increasing test prompt length decreases OOD performance; **(Right:)** increasing model size decreases OOD performance. The figures show that Transformers transcend implicit Bayesian inference for the interpoaltion setting (when p(head) $\in [0.1, 0.6]$), whereas they collapse to one components (thus, do IBI) for the extrapolation case (when p(head) is either below 0.1 or above 0.6)

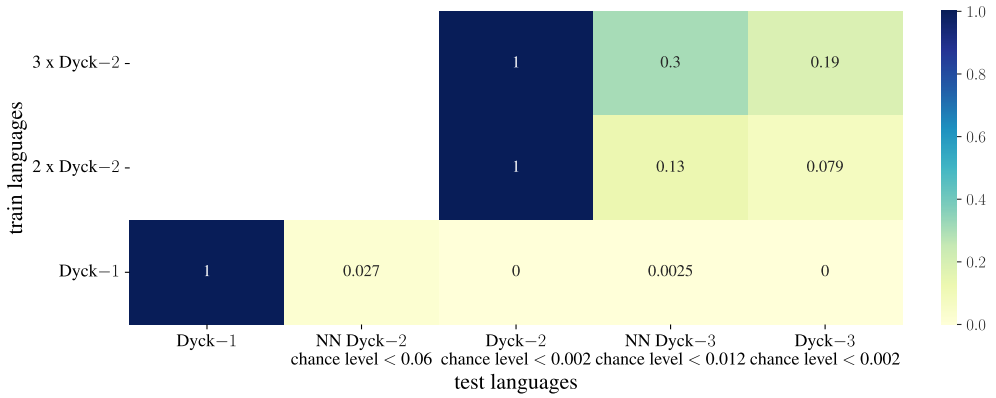


Figure 7: **Measuring transcendence on Dyck languages in Transformers, mean accuracies (across 10 seeds):** We train Transformers on simpler Dyck languages ($y-$axis, then test whether the models can finish test prompts for more complex (OOD) languages. Including more structure in the training language components (i.e., generalizing from Dyck-2 to Dyck-3) is easier for the Transformer than generalizing from simpler Dyck-1 language components

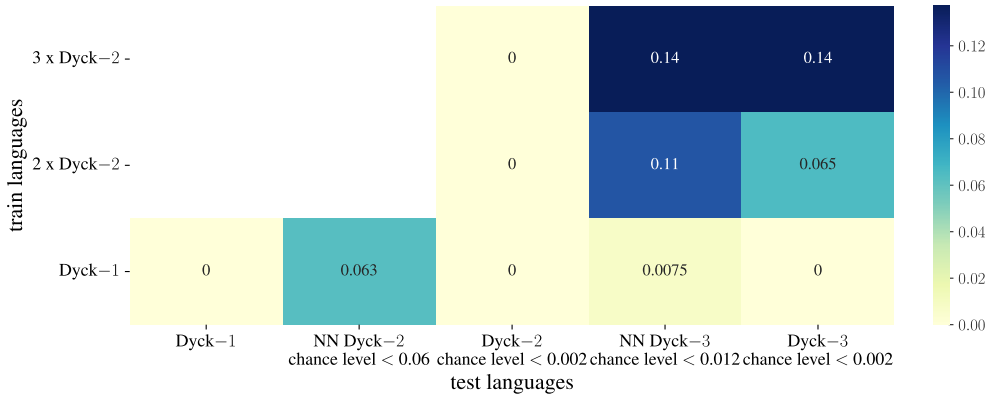


Figure 8: **Measuring transcendence on Dyck languages in Transformers, standard deviations (across 10 seeds):** We train Transformers on simpler Dyck languages ($y-$axis, then test whether the models can finish test prompts for more complex (OOD) languages. Including more structure in the training language components (i.e., generalizing from Dyck-2 to Dyck-3) is easier for the Transformer than generalizing from simpler Dyck-1 language components

Appendix D. Acronyms

CE cross entropy

AR LM autoregressive language model

IBI implicit Bayesian inference
ICL in-context learning

LM language model

OOD out-of-distribution