IMPLICIT BAYESIAN INFERENCE IS AN INSUFFICIENT EXPLANATION OF LANGUAGE MODEL BEHAVIOR IN COMPOSITIONAL TASKS

Szilvia Ujváry[1], **Anna Mészáros**[1], **Wieland Brendel**[2,3,4], **Patrik Reizinger**[*2,3], **and Ferenc Huszár**[*1]

[1]University of Cambridge, Cambridge, United Kingdom
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[3]ELLIS Institute Tübingen, Tübingen, Germany
[4]Tübingen AI Center, Tübingen, Germany

ABSTRACT

Apparently rational behaviors of autoregressive LLMs, such as in-context learning, have been attributed to implicit Bayesian inference (IBI): since training data is best explained as a mixture, the optimal next-token-predictor learns to implicitly infer latent concepts and completes prompts consistently with Bayesian inference. While the optimal strategy in-distribution, Bayesian inference is generally suboptimal on out-of-distribution (OOD) prompts due to model misspecification. As model behavior on OOD prompts is only weakly constrained by pretraining, it is not guaranteed that Bayesian behavior is extrapolated OOD. Our work investigates with small-scale experiments the degree to which Bayesian inference remains a good model of LM behavior on OOD prompts. We report two findings: (1) Transformers are less prone to collapsing into a single mixture component than Bayesian inference. Like tempered Bayesian inference, this may be advantageous under model misspecification. (2) Transformers can generalize compositionally, even when the Bayes posterior is undefined. We conclude that autoregressive LMs can display rational-looking behavior that cannot be explained as any form of generalized Bayesian inference using only the training data.

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., 2022a; 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). IBI posits that models trained on mixture data infer concepts from in-distribution prompts and complete test prompts consistently with the Bayesian posterior predictive. Xie et al. (2022a); Wang et al. (2024c) explained ICL, whereas Dalal & Misra (2024) more general language model (LM) inference via IBI. The Bayesian view is advantageous as it can be loosely regarded as a form of intelligence: the posterior predictive is constructed through systematic updates and converges to the true data distribution when the Bayesian model is well-specified (Van der Vaart, 1998).

However, Bayesian models struggle to make accurate predictions under model misspecification, i.e., when the true parameter or data distribution has zero weight under the prior (Masegosa, 2020; Morningstar et al., 2022). Large Language Models (LLMs) are trained on massive mixture distributions exposing them to various tasks (e.g., books, medical texts, journalism, technical manuals). Yet it is unclear whether each user prompt is sufficiently close to the training data. Thus, LLMs might still operate under model misspecification. In that case, their success is even more remarkable, and their understanding is even less clear.

[*]Joint senior authors. Correspondence to sru23@cam.ac.uk. Code available at: github.com/szilviaujvary/llm_implicit_bayes

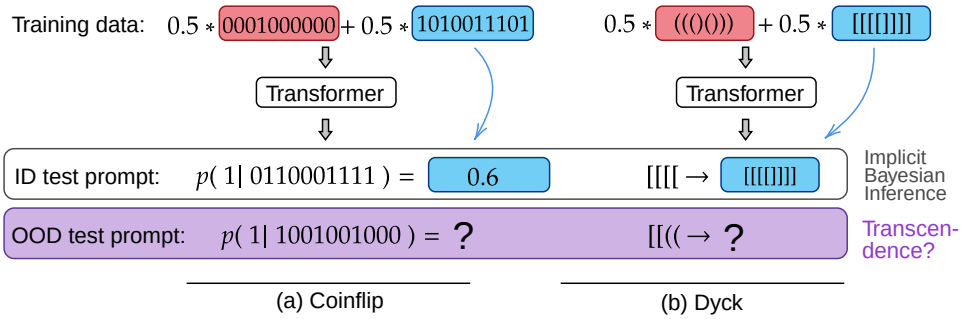

Figure 1: **Transformers can exhibit more complex behaviours out-of-distribution (OOD) on mixture data than implicit Bayesian inference (IBI) would predict:** we train Transformers on mixture data for IBI and evaluate whether they can transcend this behaviour on OOD prompts. **(Left:)** the *coinflip* experiments (1 stands for *heads* and 0 for *tails*)(§ 4.1) contain training sequences with $p(1)$ of either 0.1 or 0.6. On in-distribution (ID) test prompts, IBI predicts according to one of the mixture components, but OOD behaviour (e. g, when $p(1) = 0.3$) can differ. **(Right:)** *Dyck* experiments (§ 4.2) contain training sequences of either parentheses `()` or brackets `[]`. The Transformer completes the ID test prompts according to IBI, but OOD behaviour (e. g sequence containing both `()` and `[]`) is unconstrained by IBI

On out-of-distribution (OOD) prompts, model behavior is weakly constrained by the training data, hence behaviors beyond the Bayesian one are possible. Recent work by Wang et al. (2024b) has found that on tasks testing generalization from simple linear functions to more complex quadratic functions, ICL tends to fit linear functions based on OOD downstream task context, providing an example of extrapolating the Bayesian behavior even when it is suboptimal. In contrast, Zhang et al. (2024) showed that models can outperform (*transcend*) this behavior with majority voting enabled by low-temperature softmax: on chess tasks, LMs were able to beat all experts in the training data. However, low temperature in the softmax is only a hyperparameter, and can be implemented in (implicit) Bayesian models as well. Hence we can ask:

could there be other, more meaningful sources of transcendence over IBI in OOD tasks?

Our paper gives an affirmative answer. We showcase simple compositional tasks, where models trained for IBI exhibit more complex behavior on OOD prompts beyond majority voting. To uncover further sources of transcendence, we test on whether decoder-only Transformers (Vaswani et al., 2017) can combine the components of their training data OOD. We create OOD tasks controlling the level of model misspecification and compositionality (Fig. 1). We present two cases: coinflip data and formal Dyck languages. In coinflip data, we study how a model handles two-component mixtures with unseen head probabilities—the difficulty depends on whether the test mixture has probabilities within or outside of the interval defined by the parameters of the two training components. For Dyck languages, we investigate whether the model can compose the training components into more complex languages—the difficulty depends on the (lack of) interaction between the components (for details, cf. Tab. 3 and § 4.2). Our **contributions** are:

- Based on the limitations of Bayesian inference under model misspecification, we define three models of LLM behavior on OOD prompts, which are all approximately consistent with IBI in-distribution.
- After training small-scale Transformers to approximate IBI on the training data mixture, we test which of these behaviors best matches Transformers' predictions on OOD prompts.
- On low-probability OOD coinflips, we show that Transformers can outperform IBI via avoiding collapse into one of the training data components.
- On compositional rule-learning tasks involving Dyck$-\{i\}$ languages as training components and evaluating on more complex Dyck$-\{i+1\}$ prompts, we showcase OOD performance better than generalizations of Bayesian inference, and suggest that Transformers combine information about rules seen in separate training components.

2 Background

We study next-token prediction in autoregressive probabilistic models. Let $x_{1:L} := (x_1, x_2, \ldots, x_L)$ 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 $\{p(x_{n+1}|x_{1:n}, w);\ 1 \leq n \leq L\}$ of conditional distributions with parameters w in some fixed parameter space W. For any distribution p, let $supp(p)$ denote its support. We provide an extended overview of related work in Appx. A.

2.1 Implicit Bayesian Inference

Intuition. Implicit Bayesian inference is a property of probabilistic models. Informally, it is the ability to implicitly extract latent variables present in the training data—often interpreted as concepts (Xie et al., 2022b)—and produce next-token distributions consistent with explicit Bayesian inference over the latent variables of 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.

To formally define implicit Bayesian inference, we assume that our data can be viewed as a mixture of simpler components, i.e. $p(x_1, \ldots, x_L) = \int_\Theta p(\theta)p(x_1, \ldots, x_L \mid \theta)d\theta$, with mixing variable θ in some parameter space Θ. A Bayesian model perfectly fitting this data produces its next-token predictions according to the posterior predictive $p(x_{n+1} \mid x_n, ..., x_1) = \int_\Theta p(x_{n+1} \mid \theta, x_n, ..., x_1)p(\theta \mid x_n, ..., x_1)d(\theta)$, which marginalizes the likelihood of x_{n+1} given θ and the prompt $x_1, x_2, \ldots, x_n$ 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.

Definition 1 (Implicit Bayesian Inference (IBI))**.** *We say that a model performs implicit Bayesian inference, if after training on a mixture data distribution, its produced next-token predictions align with the next-token probabilities of the Bayesian posterior predictive over the train distribution.*

Typically, model training can be interpreted as approximating IBI. Specifically, training autoregressive models on the next-token prediction objective via minimizing cross-entropy encourages the model to perfectly fit the posterior predictive corresponding to the training mixture (if the model is expressive enough). In practice, models do not exactly reach the expected loss's minimum; they only approximate an explicit Bayesian model.

Optimality and model misspecification. 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 corresponding to the prompt is contained in Θ (Van der Vaart, 1998). However, when evaluated on out-of-distribution prompts from components outside of Θ, the Bayesian posterior predictive is no longer asymptotically optimal, and neither is IBI. Although minimal distribution shift can be tolerated in some cases (Xie et al., 2022a), even extensive training data cannot cover all distributions, leading to some degree of model misspecification. Therefore, it is interesting to ask whether language models approximating IBI in-distribution can transcend Bayesian models in tasks where they are misspecified. In this paper, we test this on synthetic datasets with varying levels of model misspecification.

2.2 Formal Languages

Formal languages are abstractions of natural languages, consisting of sequences of symbols from a fixed alphabet that obey a set of rules called formal grammar. Although much simpler than natural languages, their clear rules allow for a systematic study of OOD generalization. Moreover, as programming and mathematical theorem-proving languages are also formal languages, studying this family has real-world implications for code generation and mathematical reasoning in LLMs.

Dyck languages. 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 are in Tab. 2 in Appx. D.2. 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. NN Dyck languages are context-sensitive, which means that they depend on the position in the sequence, and are therefore considered harder in the Chomsky hierarchy. However, context-sensitive languages have more lenient rules as they don't require correct nesting, and hence are often easier to learn with language models (Mészáros et al., 2024). Our work uses Dyck languages to test whether models trained on mixtures of simpler Dyck languages can extrapolate on more complex versions.

3 THREE MODELS FOR LANGUAGE MODEL BEHAVIOR ON OOD PROMPTS

IBI is only enforced on the training distribution, leaving other behaviors possible on OOD prompts. Here we describe formally when deviation from a Bayesian model is possible OOD, and propose extensions of IBI as hypotheses for explaining language models' behaviour OOD. Our experiments in § 4 test these behaviors. We motivate our approach with an example.

Example: Mixture of coinflips. Suppose p_{train} is a mixture of i.i.d. Bernoulli distributions, i.e.,

$$p_{\text{train}}(x_{1:n}) = 0.5Bern(x_{1:n}, \theta_1) + 0.5Bern(x_{1:n}, \theta_2),$$

where $Bern(x, \theta)$ is the probability mass function of a Bernoulli random variable with parameter θ. We consider two models:

- *Tempered Bayesian inference*: defined as $p_{\text{tempBayes}}(1|x_{1:n}) := p_{\text{temp}}(\theta_1|x_{1:n}) \cdot 0.6 + p_{\text{temp}}(\theta_2|x_{1:n}) \cdot 0.1$, where the tempered posteriors p_{temp} are computed by raising the component likelihoods to $1/t$, where $t > 1$.
- *Generalist Model:* defined as $p_{\text{gen}}(x_{n+1} = 1|x_1, \ldots, x_n) = \sum_{i=1}^{n} x_n/n$, framing the two Bernoulli components as instantiating a more general pattern: that the ratio of 0's and 1's in each prompt is approximately constant.

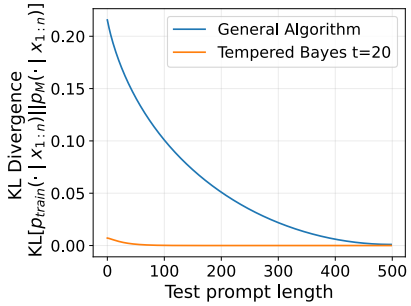

Figure 2: **Longer test prompts make tempered Bayes and the general algorithm consistent with IBI on coinflip data:** refer to the text on the left for details

A simulation in Fig. 2 shows that with increasing test prompt length both models become consistent with IBI on the train data distribution, but they behave differently OOD: tempered Bayes does concentrates on one of the training components less than regular Bayes, and the generalist model approximates $Bern(\theta = \sum_{i=1}^{n} x_i/n)$ draws.

In the rest of this section, we assume that an autoregressive model $p_M(x_{n+1}|x_{1:n})$ has been trained on a mixture distribution $p_{train}(x_{1:n}) = \sum_{i=1}^{k} p_i(x_{1:n})p(i)$ with component likelihoods $p_i(x_{1:n})$ and component weights $p(i)$. A straightforward extension of IBI to OOD data is to say that the model does IBI in-distribution, and then re-uses the same mixture components OOD, which we call:

Definition 2 (Extrapolated Bayesian inference)**.** *An autoregressive model $p_M(x_{n+1}|x_{1:n})$ performs extrapolated Bayesian inference on test distribution $p_{test} \neq p_{train}$ if $p_M(x_{n+1}|x_{1:n})$ approximates the Bayesian predictor corresponding to p_{train} well, i.e., $\forall x_{1:n} \sim p_{test}$ there exists a decomposition such that, $p_M(x_{n+1}|x_{1:n}) \approx \sum_{i=1}^{k} \omega_i(x_{1:n})p_i(x_{n_1}|x_{1:n})$, where ω_i is the Bayesian posterior $p(i|x_{1:n})$ and $p_i(x_{n+1}|x_{1:n})$ is the predictive likelihood of component i.*

For practicality, we allow approximate equality to the Bayesian predictor—the degree of equality can be picked for specific tasks. To describe the settings in which extrapolated Bayesian inference happens, we make the following observation.

Observation 1. *If $supp(p_{test}) \subseteq supp(p_{train})$, then every model achieving zero KL divergence from the train data distribution on ID test prompts, i.e., $KL[p_{train}(x_{n+1}|x_{1:n})||p_M(x_{n+1}|x_{1:n})] = 0$ must perform extrapolated Bayesian inference. If $supp(p_{test}) \not\subseteq supp(p_{train})$, a model can achieve zero KL divergence without performing extrapolated Bayesian inference.*

Even when $supp(p_{test}) \subseteq supp(p_{train})$, practical models do not typically reach zero KL divergence. If we relax the condition to KL $< \epsilon$ for some arbitrary $\epsilon > 0$, then behaviors other than extrapolated BI are possible. Hence we define two further extensions of IBI to OOD data. These behaviors can all approximate IBI on the train data distribution, but can be better strategies OOD.

Definition 3 (Generalized Bayesian Inference)**.** *We say that an autoregressive model $p_M(x_{n+1}|x_{1:n})$ performs generalized Bayesian inference on test distribution $p_{test} \neq p_{train}$ if $\forall x_{1:n} \sim p_{train}, p_M(x_{n+1}|x_{1:n}) \approx p_{train}(x_{n+1}|x_{1:n})$, and there exists a decomposition of p_M into a mixture model such that $\forall x_{1:n} \sim p_{test}, p_M(x_{n+1}|x_{1:n}) \approx \sum_i \tilde{\omega}_i(x_{1:n})\tilde{p}_i(x_{n+1}|x_{1:n})$.*

Generalized Bayesian inference is more general than extrapolated Bayesian inference as it admits a general notion of component mixing in the predictor: (1) the weights $\tilde{\omega}_i(x_{1:n})$ may differ from the Bayes posterior such as in our tempered Bayesian inference example, and (2) the predictive likelihoods

$\tilde{p}_i(x_{n+1}|x_{1:n})$ may have larger support, potentially defining completions for zero-probability prompts under p_{train}. Similar definitions exist in PAC-Bayesian contexts (Alquier, 2024).

Definition 4 (Generalist Model)**.** *An autoregressive model $p_M(x_{n+1}|x_{1:n})$ is a generalist model if $\forall x_{1:n} \sim p_{train}, p_M(x_{n+1}|x_{1:n}) \approx p_{train}(x_{n+1}|x_{1:n})$, but the model $p_M(x_{n+1}|x_{1:n})$ captures a more general pattern in the data rather than approximating each component of p_{train}.*

What constitutes a "more general pattern in the data" can depend on the task. Intuitively, Generalized Bayesian models are an interpolation between extrapolated BI and Generalist models, based on how much the individual component predictions are aligned. That is, how much information is shared between the components. A Generalist Model shares most information, since it frames all training components as instances of the same pattern. The following observation shows that alignment of the component predictives (which we call "information sharing") can be beneficial OOD.

Observation 2. *[Information sharing between components helps OOD] Suppose we have a Generalized Bayesian model, i.e., $p_M(x_{n+1}|x_{1:n}) = \sum_{i=1}^{k} \tilde{\omega}_i(x_{1:n})\tilde{p}_i(x_{n+1}|x_{1:n})$, and $\forall i : supp(p_{test}) \subseteq supp(\tilde{p}_i)$. Then,*

$$KL[p_{test}(x_{n+1} \mid x_{1:n})||p_M(x_{n+1} \mid x_{1:n})] \leq \min_j \{KL[p_{test}(x_{n+1} \mid x_{1:n})||\tilde{p}_j(x_{n+1} \mid x_{1:n})]\} + \mathbb{E}_{x_{1:n+1}}\left[\sum_{i=1}^{k} \tilde{\omega}_i(x_{1:n}) \log \frac{\tilde{p}_j(x_{n+1} \mid x_{1:n})}{\tilde{p}_i(x_{n+1} \mid x_{1:n})}\right].$$

The proof and an illustrative example for Dyck languages is in Appx. B. Observation 2 shows that information sharing, quantified by the log of the prediction ratios between components, can be beneficial because it helps bound the KL divergence between the (predictive) model and the OOD test distribution better.

4 Experimens

Setup. Throughout this work, we focus on decoder-only Transformers. 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). Complete details of our experimental setup are in Appx. D.

4.1 Extrapolation to OOD Coinflips: Transcendence via Generalized BI

Coinflip Datasets and Tasks. We train on a mixture of coinflips: two i.i.d. Bernoulli components with success (head) 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 and have a different head probability, making draws from most test distributions very unlikely under the train data. The test prompts are drawn i.i.d. from:

1. **Interpolation:** a single Bernoulli with success probability $\theta \in [0.1, 0.6]$ (Fig. 3)
2. **Extrapolation:** a single Bernoulli with success probability $\theta \in [0, 0.1) \cup (0.6, 1.0]$ (Fig. 3)
3. **Markov Chain:** 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 initialize the Markov chain at its stationary distribution. (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 via the Markov chain.

Metrics. We measure the probability of predicting heads or tails as the next token, and compare to the Bayesian posterior predictive.

Results. Our results show that Transformers accurately learned the correct next-token probabilities of $p(1)$ and $p(0)$ on the train data components, i.e., when $p(1) = 0.1$ or $= 0.6$ in the test prompt, the model predicted accordingly (Fig. 3). Therefore, the Transformers closely approximate implicit Bayesian inference on the ID prompts. The performance of Transformers on **low-probability OOD coinflips** varies based on whether the OOD success probability $p(1)$ 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 slightly below Bayesian inference outside.

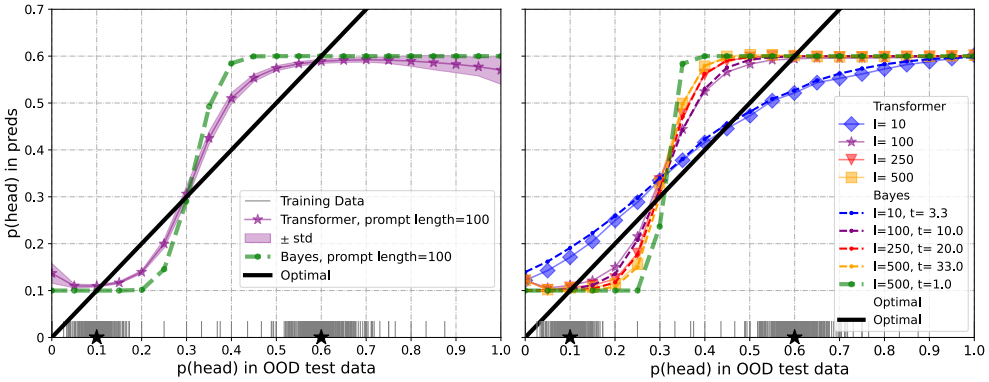

Figure 3: **Predicted head probabilities for OOD coinflip data**: Models were trained on two mixture components with $p(1)$ set to either 0.1 or 0.6. The optimal solution (the $y = x$ line) is **black**, the Bayesian estimate **green**, the component $p(1)$'s a star ★ on the $x-$axis with training samples' empirical probabilities as discrete lines. **(Left:)** comparing the Transformer to Bayesian inference on test prompts of length 100 (means and standard deviations across 10 training seeds). **(Right:)** Transformer predictions for varying test prompt lengths l and tempering t. **(Conclusion:)** Transformers closely match tempered Bayesian inference. This implicit tempering allows Transformers to beat Bayesian inference within $[0.1, 0.6]$ as they do not collapse into one training component and stay closer to the optimal **black** line.

Explanation: Transformers implicitly temper the likelihood. To see what causes the superior performance within $[0.1, 0.6]$, we ablated the test prompt length from 10 to 500 Fig. 3 (right). This revealed that Transformers' predictions match Bayesian inference much better if we introduce *tempering* into the Bayesian likelihood, i.e. the likelihood becomes $L(x) = p(x \mid \theta)^{1/t}$. Temperature values $t > 1$ slow down the posterior's convergence by weakening the likelihood, providing a more balanced weighting of train components within $[0.1, 0.6]$. This is a useful approach to mitigate model misspecification, as such models are less prone to collapsing into a single training data component. To see how the models achieve this tempered IBI approximation, we ablated the architecture by canceling out the effects of individual attention heads via replacing their outputs on test prompts with random noise, a technique known in the mechanistic interpretability literature as *knockout* (Rai et al., 2025). Interestingly, the resulting curves were similar to the curves using shorter prompt lengths in Fig. 3, which suggests that the implicit tempering effect is better viewed as Bayesian prediction based on smaller segments of the test prompts. Or equivalently, as tempering with large temperature (both effects divide the log-likelihood by a large positive number). This can be interpreted as the attention heads learning to focus on small segments of the test prompts (i.e., having small "attention span"), which results in slower posterior convergence. The complete description of this ablation and its results (Fig. 7) along with further ablations on the architecture can be found in Appx. D.1.

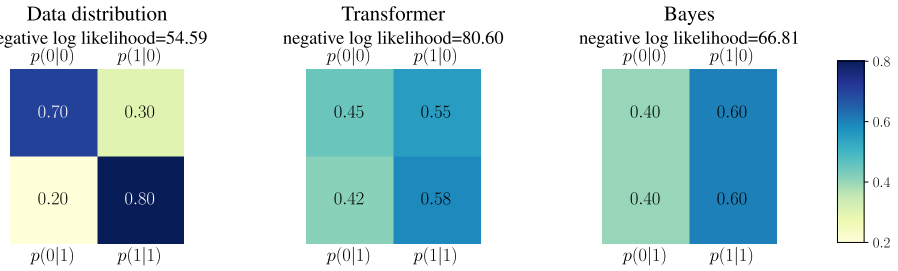

Figure 4: **Transition probabilities and negative log-likelihood (nll ↓) on OOD Markov chain data:** Models were evaluated on prompts drawn from a Markov chain with transition probabilities **(Left)**, the chain was initialized at its stationary distribution p(head)$= 0.6$. The Transformers' transition probabilities **(Center)** were estimated from the next-token probabilities based on the previous token. Compared to the Bayes model **(Right)**, where each token is independent, the Transformer's transition matrix is slightly closer to the ground-truth transitions, but the nll is much higher.

The Transformer's estimated transition matrix on **Markov chain data** is slightly closer to the true transitions than those of a Bayesian model (Fig. 4). This indicates that the Transformer captured in

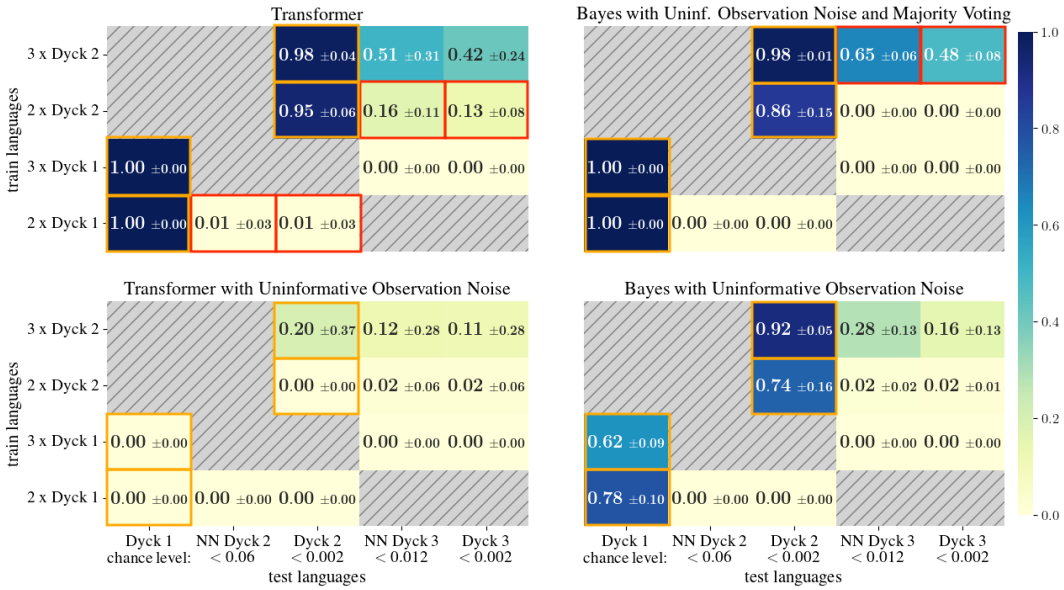

Figure 5: **The role of information sharing in Transformers and Bayesian inference on Dyck languages:** We train Transformers **(Left)** on simpler Dyck languages ($y-$axis), then test whether the models can finish test prompts for more complex (OOD) languages ($x-$axis) and compare it to Bayesian inference with **(Top Right)** and without **(Bottom Right)** majority voting. We add uninformative observation noise (via defining random outputs for unseen tokens) to all but the vanilla Transformer **(Top Left)**. ID tasks are marked in **orange**. The best result in each OOD task is highlighted in **red**. **(Conclusion:)** 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. Means and standard deviations were computed from at least 4 seeds

its predictions only a limited amount of the autocorrelations seen only in the test prompts. Hence this task proved to be too difficult for Transformers.

Summary of Results Transformers are less prone to collapsing to a single component than IBI, but this beneficial implicit tempering effect may be due to a limited attention span. When tested on Markov chain data, Transformers struggle with extrapolation.

4.2 Rule-learning: Transcendence via Generalized BI and Information Sharing

Dyck language Datasets and Tasks. The training data comes from mixtures of Dyck$-i$ ($i \in \{1, 2\}$) languages, containing either two or three components, depending on the task—the difference between these components is the type of parentheses we use. Each component shares the same depth i. The test prompts are from a Dyck$-\{i+1\}$. 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 `SOS ( { [ { ( { } ) }`. We include all of our experimental scenarios in Tab. 3 in Appx. D.2.

Quantifying the compositionality of our tasks. 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. For example, a model trained on a mixture of Dyck-1 `(` and Dyck-1 `[` only needs to follow both rules simultaneously to complete prompts obeying NN Dyck-2 `(`, `[`, but needs to explicitly prefer the idea of *nesting* two types of brackets when it's evaluated on Dyck-2 `(`, `[`. Therefore, although both tasks are OOD, the first one requires a simpler, more general notion of composition than the second. We provide a simple, heuristic definition for quantifying the compositionality of our tasks in Appx. C, and present the computed values in Tab. 3.

Defining a Generalized Bayes Model. Since all test prompts contain unseen tokens and thus have zero likelihood under each individual train component, a Bayesian posterior cannot be defined in these tasks. However, we can test for the presence of generalized Bayesian inference via adding

uninformative observation noise for the unseen token to each component likelihood in the Bayesian model. Specifically, we augment component likelihoods so that, conditioned on any prompt, the unseen tokens have constant probability 0.005 as next-tokens. Since our Transformers use greedy decoding, which is equivalent to majority voting (Zhang et al., 2024), we also implement the Bayesian model with majority voting. More details and ablations on the models' construction are provided in Appx. D.

Metrics. We evaluate grammatical accuracies in both the train and test grammars, assessing the full, completed prompt in both Transformers and generalized Bayesian models. That is, we check whether all grammar rules of the train/test languages hold. When evaluating in the OOD settings, we also 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. .

Results. Transformers approximate IBI on the training data well, as indicated by Fig. 5. Transformers beat the generalized Bayes model without majority voting on all OOD tasks, but underperform generalized Bayes with majority voting when trained on 3 Dyck 2 components. This indicates that this generalized Bayesian model can at best explain the Transformers' extrapolation only on this one task. An alternative approach (based on Observation 2) is to assume information sharing between the Transformers' component approximations: our knockout experiment (Fig. 6) confirms this since removing the effect of single attention heads does not completely remove knowledge about each train components' rules in the model. The large standard deviations observed in the Transformer results suggests that very good extrapolation is possible, but not consistently learnt.

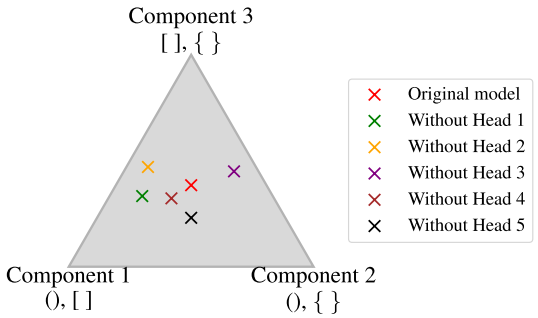

Figure 6: **Attention heads for training data components by knockout:** using models trained on 3 Dyck-2 components, we knock out attention heads and test how well the resulting models obey each Dyck 2 grammar on the OOD prompts (Dyck 3). Model accuracies on the train grammars were converted into barycentric coordinates: being close to vertex i means higher accuracy on component i. Compared to the original model (**red**), performance moving away from each component indicates specialization towards that component in the knocked-out head. For example, head 3 (**purple**) specializes in component 1, the Dyck 2- `(, [` language.

Summary of Results. In rule-based tasks, Transformers' OOD performance is better than generalized Bayes models', and beyond majority voting, this transcendence seems to result from information sharing between the train components' approximations in the model.

5 Discussion

Limitations. Our analysis is based on training small-scale models on small-scale data sets, which limits the predictive power of our conclusions—nonetheless, we still observed a non-trivial OOD behavior in Transformers. As suggested by prior works such as those on grokking (Liu et al., 2023; 2022; Wang et al., 2024a; Power et al., 2022), a qualitatively different behavior might emerge if we trained much longer. We leave this for future work.

Conclusion. We have empirically shown that when trained on data mixtures, although their training encourages implicit Bayesian inference, Transformers can meaningfully transcend this behaviour, implementing more general approximations of Bayesian inference. These approaches can transfer better to compositional OOD tasks. We have highlighted the benefits of information sharing between the train components' approximations, paving the way towards building generalist models.

Acknowledgements

This work was supported by a Turing AI World-Leading Researcher Fellowship G111021. Patrik Reizinger acknowledges his membership in the European Laboratory for Learning and Intelligent

Systems (ELLIS) PhD program and thanks the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for its support. This work was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. Wieland Brendel acknowledges financial support via an Emmy Noether Grant funded by the German Research Foundation (DFG) under grant no. BR 6382/1-1 and via the Open Philantropy Foundation funded by the Good Ventures Foundation. Wieland Brendel is a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645. This research utilized compute resources at the Tübingen Machine Learning Cloud, DFG FKZ INST 37/1057-1 FUGG.

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]. 13

Pierre Alquier. User-friendly introduction to pac-bayes bounds. *Foundations and Trends® in Machine Learning*, 17(2):174–303, 2024. ISSN 1935-8245. doi: 10.1561/2200000100. URL http://dx.doi.org/10.1561/2200000100. 5

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]. 13

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. 13

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]. 13

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]. 13

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. 3, 18

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. 1, 13

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. 13

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. 13

Javid Ebrahimi, Dhruv Gelda, and Wei Zhang. How can self-attention networks recognize dyck-n languages?, 2020. URL https://arxiv.org/abs/2010.04303. 13

OpenAI et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774. 20

Fabian Falck, Ziyu Wang, and Chris Holmes. Is in-context learning in large language models bayesian? a martingale perspective. ICML'24. JMLR.org, 2024. 13

PyTorch Lightning Falcon. Pytorch lightning: Accelerated research from prototypes to production. https://github.com/PyTorchLightning/pytorch-lightning, 2019. 15

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]. 13

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]. 13

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. 13

Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking Beyond Algorithmic Data. September 2022. URL https://openreview.net/forum?id=zDiHoIWa0q1. 8

Ziming Liu, Ziqian Zhong, and Max Tegmark. Grokking as Compression: A Nonlinear Complexity Perspective, October 2023. URL http://arxiv.org/abs/2310.05918. arXiv:2310.05918 [cs, stat]. 8

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. URL http://arxiv.org/abs/1711.05101. arXiv:1711.05101 [cs, math]. 20

Andres R. Masegosa. Learning under model misspecification: Applications to variational and ensemble methods, 2020. URL https://arxiv.org/abs/1912.08335. 1, 13

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. 13

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. 1

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. 13

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 (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 8270–8298. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/morningstar22a.html. 1, 13

Anna Mészáros, Szilvia Ujváry, Wieland Brendel, Patrik Reizinger, and Ferenc Huszár. Rule extrapolation in language models: A study of compositional generalization on ood prompts, 2024. URL https://arxiv.org/abs/2409.13728. 3, 5, 13

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of transformers with simple arithmetic tasks, 2021. 13

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. 1

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. 15

Steven Pinker. *The language instinct*. William Morrow, New York, NY, December 1994. 13

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]. 8

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245. 20

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. pp. 24, 2018. 20

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models, 2025. URL https://arxiv.org/abs/2407.02646. 6

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]. 13

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. 13

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]. 13

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. 13

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 (eds.), *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pp. 44–54, Florence, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3905. URL https://aclanthology.org/W19-3905/. 13

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. 1

A. W. Van der Vaart. "10.2 bernstein–von mises theorem", asymptotic statistics. *Cambridge University Press*, 1998. 1, 3

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. 2, 20

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked Transformers are Implicit Reasoners: A Mechanistic Journey to the Edge of Generalization, October 2024a. URL http://arxiv.org/abs/2405.15071. arXiv:2405.15071 [cs]. 8

Qixun Wang, Yifei Wang, Yisen Wang, and Xianghua Ying. Can in-context learning really generalize to out-of-distribution tasks?, 2024b. URL https://arxiv.org/abs/2410.09695. 2, 13

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, 2024c. URL https://arxiv.org/abs/2301.11916. 1, 13

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. 1

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. 13

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]. 13

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]. 13

Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc. 13

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. 15

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference, 2022a. 1, 3, 13

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An Explanation of In-context Learning as Implicit Bayesian Inference, July 2022b. URL http://arxiv.org/abs/2111.02080. arXiv:2111.02080 [cs]. 3

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]. 20

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. 2, 8, 19

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]. 1

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. 13

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). Most works focus on explaining in-context learning as IBI. Although we don't explicitly consider in-context learning tasks, this line of work is very much relevant to us, since our coinflip experiments can be rephrased as in-context learning tasks (due to using long test prompts). Xie et al. (2022a) show in a theoretical setting that, on a mixture of HMMs data, IBI over the components achieves in-context learning. Wang et al. (2024c) give a similar interpretation for ICL, calling LLMs implicit topic models and providing evidence for real-world LLMs, while Wies et al. (2023) gives learnability conditions via PAC-Bayes. Wang et al. (2024b) demonstrates that in in-context learning can fail to generalize OOD due to collapsing into one of the train components. The most relevant for us from their tested settings is generalization from simple linear functions to more complex quadratic ones. In light of our results finding nontrivial deviation from IBI, their OOD tasks seem too different from the training tasks, and require much more complex compositions of training components than our tasks. There are also 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 & 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 to Mészáros et al. (2024), who study rule extrapolation, the ability to complete prompts with partly broken rules when trained on formal languages. Transformers excel in 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 & Mansouri, 2024; Han & Padó, 2024; Ramesh et al., 2024; Lake & 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 & 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).

B Proof and discussion of Observation 2

Observation 2. *[Information sharing between components helps OOD] Suppose we have a Generalized Bayesian model, i.e.,* $p_M(x_{n+1}|x_{1:n}) = \sum_{i=1}^k \tilde{\omega}_i(x_{1:n})\tilde{p}_i(x_{n+1}|x_{1:n})$*, and* $\forall i : supp(p_{test}) \subseteq supp(\tilde{p}_i)$*. Then,*

$$KL[p_{test}(x_{n+1} \mid x_{1:n})||p_M(x_{n+1} \mid x_{1:n})] \leq \min_j \{KL[p_{test}(x_{n+1} \mid x_{1:n})||\tilde{p}_j(x_{n+1} \mid x_{1:n})]\} + \mathbb{E}_{x_{1:n+1}}\left[\sum_{i=1}^k \tilde{\omega}_i(x_{1:n}) \log \frac{\tilde{p}_j(x_{n+1} \mid x_{1:n})}{\tilde{p}_i(x_{n+1} \mid x_{1:n})}\right].$$

Proof. For ease of notation, we leave out the conditionals' arguments, i.e., we denote $p_{\text{test}}(x_{n+1} \mid x_{1:n})$ as p_{test} with the understanding that statements like $x_{1:n} \sim p_{\text{test}}$, and $x_{n+1} \sim p_{\text{test}}$ mean $x_{1:n} \sim p_{\text{test}}(x_{1:n})$ and $x_{n+1} \sim p_{\text{test}}(x_{n+1} \mid x_{1:n})$, respectively. Then,

$$\begin{aligned}
KL[p_{\text{test}}||p_M] &= \mathbb{E}_{x_{1:n}\sim p_{\text{test}}}\left[\mathbb{E}_{x_{n+1}\sim p_{\text{test}}}\left[\log \frac{p_{\text{test}}}{\sum_{i=1}^k \omega_i(x_{1:n}) \cdot p_i}\right]\right] && (1)\\
&= \mathbb{E}_{x_{1:n+1}\sim p_{\text{test}}}\left[\log \frac{p_{\text{test}}}{p_j} \cdot \frac{p_j}{\sum_{i=1}^k \omega_i(x_{1:n}) \cdot p_i}\right] && (2)\\
&= \mathbb{E}_{x_{1:n+1}\sim p_{\text{test}}}\left[\log \frac{p_{\text{test}}}{p_j} + \log \frac{p_j}{\sum_{i=1}^k \omega_i(x_{1:n}) \cdot p_i}\right] && (3)\\
&= KL[p_{\text{test}}||p_j] + \mathbb{E}_{x_{1:n+1}\sim p_{\text{test}}}\left[\log \frac{p_j}{\sum_{i=1}^k \omega_i(x_{1:n}) \cdot p_i}\right] && (4)\\
&\leq KL[p_{\text{test}}||p_j] + \mathbb{E}_{x_{1:n+1}\sim p_{\text{test}}}\left[\sum_{i=1}^k \omega_i(x_{1:n}) \log \frac{p_j}{p_i}\right], && (5)
\end{aligned}$$

where the last inequality is due to Jensen's inequality. This equation is true for all components p_j of p_M, hence it is true for the one achieving minimal KL divergence from p_{test}. Due to our condition on the supports, none of the terms on the RHS are infinite. We focus on this component as this is the one a Bayesian weighting of components would select most confidently. However, how much other components' predictions are aligned to this component matters for predictive performance so long as these components have nonzero weight. □

Observation 2 shows that alignment of the component predictives (which we call "information sharing") helps us bound the KL divergence between the (predictive) model and the OOD test distribution better. Therefore, high information sharing (that is, a low value for the second term in the bound) helps us guarantee a better predictive model for the OOD task at hand.

Example (Dyck languages). Suppose the training distribution is a mixture of two Dyck 1 languages, i.e. p_1 :=Dyck 1-`(, )` and p_2 :=Dyck 1-`[, ]` respectively, and the test language is Dyck 2 containing both bracket types. Let us consider the test prompt `( [`, which has zero probability under the training distribution, as the two types of opening tokens can never occur together under the training distribution. Suppose we have two models that implement (implicit) generalized Bayesian inference: Model 1 and Model 2. Model 1 approximates p_1 and p_2 with components $\tilde{p}_1$ and $\tilde{p}_2$ that have support on all tokens, but the information stored about the unseen tokens is random and does not respect the tokens' rules, e.g. $p_1(\cdot \mid$ `( [` $)$ puts nonzero weight on invalid next-token `)`. Suppose that Model 2's components approximating p_1, p_2, $\bar{p}_1$ and $\bar{p}_2$ have incomplete information about the rules of the unseen tokens. Example next-token distributions for prompt `( [` are given in Appx. B. The only invalid next-token is `)` (corresponding cells shaded in **red**). Model 2 mixes components giving lower probability to the invalid next-token than Model 1, hence on average, Model 2 performs better.

C Measuring Formal Language Task Compositionality

In this section, we present a heuristic measure for the compositionality of our Dyck language extrapolation tasks.

Component	(	)	[	]
p_1 (Train distribution)	Undefined	Undefined	Undefined	Undefined
p_2 (Train distribution)	Undefined	Undefined	Undefined	Undefined
$\tilde{p}_1$ (Model 1)	0.495	0.495	0.005	0.005
$\tilde{p}_2$ (Model 1)	0.005	0.005	0.495	0.495
$\bar{p}_1$ (Model 2)	0.6	0.1	0.1	0.2
$\bar{p}_1$ (Model 2)	0.004	0.001	0.495	0.495
Dyck 2 (Test distribution)	1/3	0	1/3	1/3

Table 1: Example next-token distributions for prompt `( [` of the components of the models considered. Cells corresponding to the only invalid next-token, `)` are shaded in **red**). Model 2 mixes components giving smaller probabilities to the invalid next-token, hence performs better.

Definition 5 (Task Compositionality)**.** *Consider an OOD task where a model was trained on the mixture of languages $L_1, L_2, \ldots, L_m$ which have no conflicting rules (that is, there exist sequences simultaneously satisfying all rules of these languages), and tested on another language L_{m+1}, using test prompts $x_{1:k} \in \mathcal{D}_{test}$. Then the compositionality c of this task is defined as*

$$c = \frac{|\{x_{1:n} \mid n > k, x_{1:k} \in \mathcal{D}_{test} \text{ and } x_{1:n} \text{ satisfies the rules of } L_1, L_2, \ldots, L_m \text{ and } L_{m+1}\}|}{|\{x_{1:n} \mid n > k, x_{1:k} \in \mathcal{D}_{test} \text{ and } x_{1:n} \text{ satisfies the rules of } L_1, L_2, \ldots \text{ and } L_m\}|}.$$

Task compositionality measures the degree of information contained in the rules of all training languages towards the test language as it quantifies how easy to guess correct test completions based on all the constraints the training components' rules place on the possible completions. $c = 1$ indicates full compositionality, while $c = 0$ means complete lack of compositionality.

The task compositionality values of our Dyck language tasks are reported in Tab. 3. Their calculation is simple, but we include an example here for completeness.

Example 1: Task compositionality for not nested test languages. These tasks all have task compositionality $= 1$, because satisfying the rules of the (nested) train languages implies pairing all bracket types in the test prompts. Since this is all the not nested language rules require, task compositionality is 1.

Example 2: Task compositionality of the Dyck 2 – `( [`, Dyck 2 – `( {` $\rightarrow$ Dyck 3 – `( [ {` task. Our OOD test prompts have the form [3 different open tokens] + [6 tokens forming a valid instance of Dyck 3]. Since the latter part is a valid sequence, we may ignore it for our calculation and focus on the three opening tokens instead. These tokens `(, [` and `{` have 6 permutations. For simplicity, we only consider completions of length 3: this is how many tokens are required to close all opening brackets. In the case of `{ ( [` and `[ ( {`, a model nesting both of the pairs `(, [` and `(, {` correctly must finish both prompts correctly, as there are no completions where both of the above token pairs are nested correctly, but the tree tokens are not. In the other four permutations, there exists exactly one correct and one incorrect completion for the test language, consistent with the training rules. For example, for `{ [ (`, the completion `) } ]` is wrong, but the pairs `(, [` and `(, {` are both nested correctly. The correct completion is `) ] }`. Therefore, $c = (2/6) \cdot 1 + (4/6) \cdot 0.5 = 2/3$.

D 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.

D.1 Coinflip Experiments

i.i.d. coinflip data (Fig. 3). 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 varying length drawn randomly between 1 and 256. The test prompts (excluding the length ablation) have length 100 and are Bernoullis drawn i.i.d. with success probabilities ranging between 0 and 1.

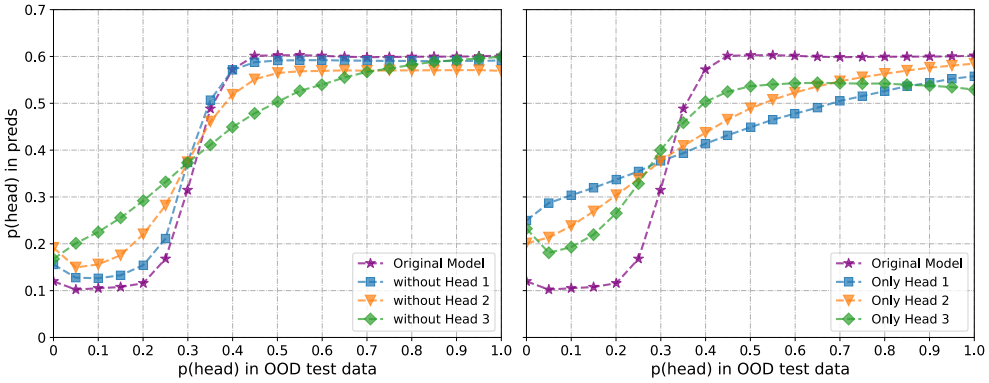

Figure 7: **The different "attention spans" of attention heads for OOD coinflip data:** prompt length is 500 and attention head outputs are randomized (knocked out) **(Left:)** one out of three heads knocked out, **(Right:)** two out of three heads knocked out. **(Conclusion:)** Similarity to tempered Bayes and shorter prompt length curves in Fig. 3 suggests that heads have differing "attention spans". For example, without *Head 3*, the model is close to the full transformer on an OOD prompt of length only 10 (blue dashed curve in Fig. 3 right).

Markov chain data (Fig. 4). For the Markov chain experiment, the training data was i.i.d. Bernoullis exactly as above. The test prompts were sampled i.i.d from 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 distribution with success parameter 0.6. The transition matrix $p(T|T) = 0.7, p(H|H) = 0.8$ was used, consistently with the desired stationary distribution. We initialized the Markov chain at its stationary distribution.

Metrics. We measure the probability of predicting heads or tails as the next token, given the test prompt. We use softmax decoding with temperature 1. The next-step probabilities computed at each step of the test prompts' completion were averaged. For the Markov chain experiment, the quantities $p(T|T), p(H|T), p(T|H), p(H|H)$ were computed by averaging the next-step predictions where the last token in the conditioned prompt was Heads and Tails, respectively.

Ablations. The experimental details of our ablation experiments are as follows:

- **Test prompt length ablation (Fig. 3, Fig. 8):** We evaluated Transformers on test prompts of length up to 500. The reported values are averages across all checkpoints. We created the curves corresponding to the varying test prompt lengths $l \leq 500$ by considering only the first l tokens of each test prompt. The Bayesian and tempered Bayesian results were generated by performing explicit (tempered) Bayesian inference on the same train data as used for the Transformer checkpoints.
- **Knockout experiments (Fig. 7, Fig. 9):** We randomly picked a checkpoint from the coinflip experiments. We augmented the Transformer decoder architecture by replacing the outputs of attention heads on each test prompt with random $N(0, 1)$ noise. We use the same test prompts as in the prompt length ablation experiment. We performed two versions: replacing the outputs of only one attention head, and every attention head except one. All other hyperparameters in the architecture and training were unchanged.
- **Ablations on the attention head structure (Fig. 8, Fig. 9):** We trained a Transformer with only one attention head, and a version where attention heads were averaged instead of concatenated in the architecture. We repeated the test prompt length ablation for both, and the attention head knockout experiment for the averaged heads version. All other hyperparameters in the architecture and training were unchanged.

Additional Results. Our ablations on the attention head structure produces similar results as the original models in which attention heads were concatentated (Fig. 8, Fig. 9). The main difference was that he one attention head and averaged attention heads setups required slightly higher temperatures to match tempered Bayesian inference, indicating that the models' attention spans were even smaller (Fig. 8).

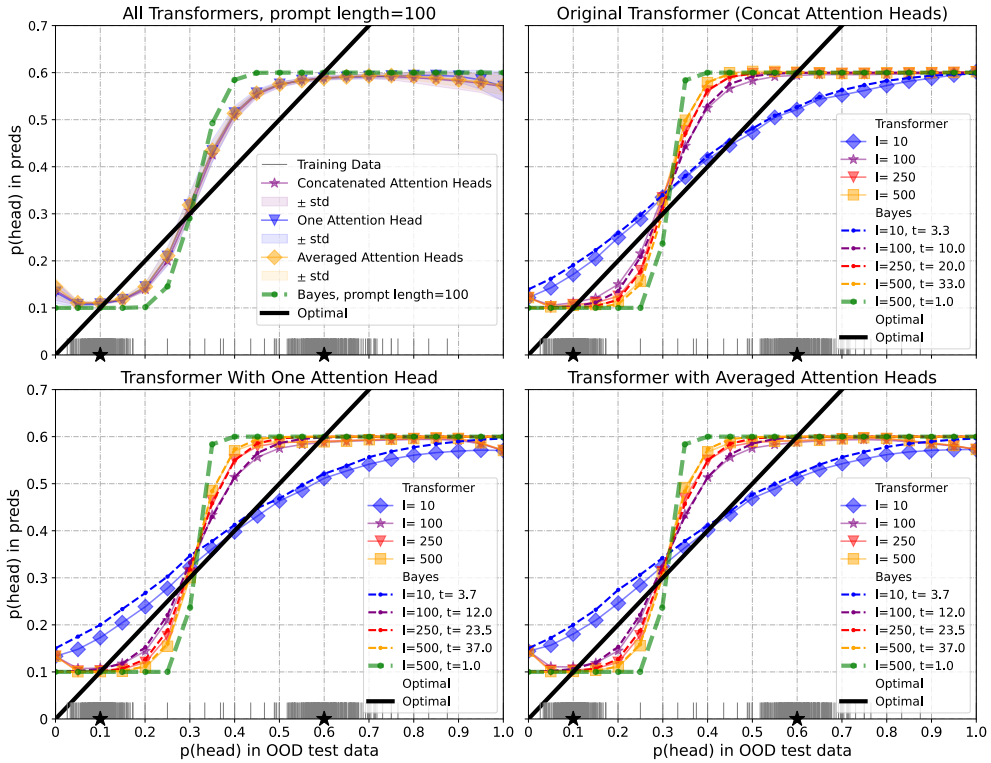

Figure 8: **Predicted head probabilities for OOD coinflip data ablating the architecture and test prompt lengths**: Models were trained on two mixture components with $p(head)$ set to either 0.1 or 0.6. The optimal solution (the $y = x$ line) is **black**, the Bayesian estimate **green**, the component $p(head)'$s a star ★ on the $x-$axis with training samples' empirical probabilities as discrete lines. **(Upper Left:)** comparison of three attention configurations in the Transformer architecture does not show significant differences; **(Upper Right and Lower Rows:)** increasing the test prompts' lengths shows the tempered Bayesian inference-like behavior for all three configurations. The one attention head and averaged attention heads setup required slightly higher temperatures to match tempered Bayesian inference.

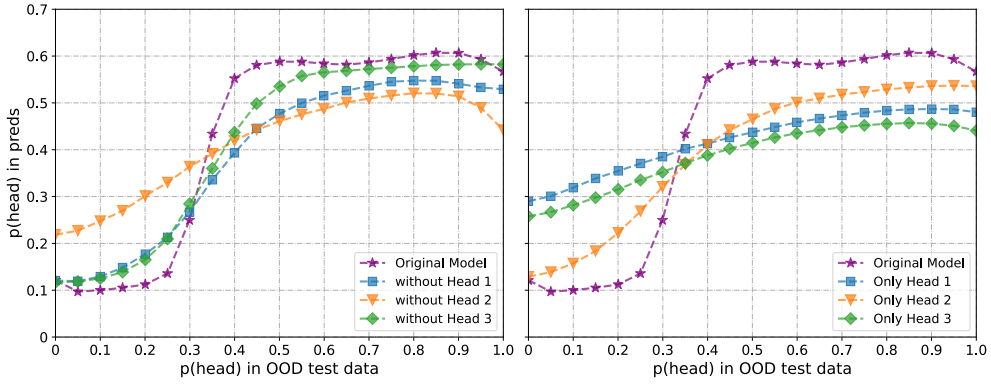

Figure 9: **Performance of Transformers with averaged attention heads on OOD prompt length 500 with attention head outputs randomized (knocked out): (Left:)** one out of three attention heads knocked out, **(Right:)** two out of three attention heads knocked out. Just like in the original Transformer models where attention heads were concatenated (Fig. 7), similarity to tempered Bayes suggests that attention heads have differing attention spans. For example, without Head 3, the model is close to the full transformer on an OOD prompt of length only 10.

D.2 DYCK LANGUAGE EXPERIMENTS

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 2: **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)

Training 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. The prior probability of each training component is equal. Within each component, we pick the proportion of bracket types so that the different bracket types appear roughly with equal probability in the full mixture. For instance, if the training components are Dyck $2-$`(, [` and Dyck $2-$`(, {`, we decrease the weight of `(`. 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.

Test data. Our in-distribution test prompts are drawn from the training distribution. The OOD test prompts are from a nested 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. Example test prompts are given in Table 4. We evaluate models on 64 ID and 64 OOD test prompts.

Metrics. We evaluated grammatical accuracies in both the train and test grammars, assessing the full, completed prompt. That is, we checked whether all grammar rules of the train/test languages hold. For the not nested (NN) languages, we computed evaluated the completed prompt with respect to the not nested grammatical rules. In these cases, the prompt itself satisfied the stricter, nested grammar rules. The same checkpoints were used to compute the nested and not nested (NN) grammatical accuracies in each row of the figures. In each training session, we saved the model achieving highest grammatical accuracy with respect to the nested language on the OOD test prompts.

Train language components			Test language	Task compositionality (Appx. C)
Dyck−2 (, [	Dyck−2 (, {	Dyck−2 [, {	NN Dyck−3 (, [, {	1
Dyck−2 (, [	Dyck−2 (, {		NN Dyck−3 (, [, {	1
Dyck−2 (, [	Dyck−2 (, {	Dyck−2 [, {	Dyck−3 (, [, {	1
Dyck−2 (, [	Dyck−2 (, {		Dyck−3 (, [, {	2/3
Dyck−1 (	Dyck−1 [		NN Dyck−2 (, [	1
Dyck−1 (	Dyck−1 [	Dyck−1 {	NN Dyck−3 (, [{	1
Dyck−1 (	Dyck−1 [		Dyck−2 (, [	1/3
Dyck−1 (	Dyck−1 [	Dyck−1 {	Dyck−3 (, [{	1/6

Table 3: **Dyck languages used for training and testing:** We list the scenarios in increasing order of task difficulty. Please refer to Tab. 2 for the description of the language rules

Test language	Example test prompt
Dyck 2 - (, [	SOS ([[( ) ( ) [ ]
Dyck 2 - (, {	SOS { [{ { ( ) } ( )
Dyck 2 - [, {	SOS [[{ { [{ }] }
Dyck 3 - (, [, {	SOS [{ (( ) [{ }]

Table 4: Example prompts for the Dyck test languages.

Selection of the Dyck checkpoints. In order to make sure our trained models stay approximate Implicit Bayesian Inference on the train components, we filtered checkpoints that achieved in-distribution test accuracy above 0.8. Out of at least 10 training seeds for each experiment, this left at least 4.

The Generalized Bayesian Models (Fig. 5, Fig. 10). We extended Implicit Bayesian Inference to zero-probability prompts. Instead of fitting a model on data, we use the data generating function used to generate the train data as the Bayesian likelihood, and perform explicit Bayesian inference with it. This is equivalent to using a model that perfectly fits the data generating distribution. In order to define a posterior on zero-probability prompts, we augment this likelihood in two ways: (1) adding uninformative (Fig. 5) and (2) informative (Fig. 10) observation noise to the each component's likelihood. *This requires modifying the likelihood of each component, so that previously unseen bracket-type tokens always have nonzero probability, i.e., they can be predicted as next-tokens, conditioned on any prompt*. In the uninformative case, the augmented probability was constant $\epsilon = 0.005$ for each unseen bracket-type token. In the informative case, the rules of the nested Dyck languages were partially broken. The unseen tokens were added to the stack (necessary for generating valid nesting) 90% of the times they were generated. To further enable incorrect prompts, invalid closure without the opening pair of the bracket at the top of the stack was given nonzero probability: for example, the model can predict } seeing prompt () [. When only a single component is present, the informative noise method achieved 90% accuracy on the rule corresponding to the unseen token, and 100% on the rules of the seen tokens. Since there was no natural way to incorporate finishing the prompt into the likelihood as the Transformer learns this from the train data lengths only, we evaluated our generalized Bayes models after 5 next-token predictions, making the completed test prompt have length 15 (including the first SOS token). This length was selected by observing that the trained Transformers have finished test prompts after 5 next-token predictions. Since our Transformers use greedy decoding, which is equivalent to zero temperature in the softmax, and thus majority voting (Zhang et al., 2024), we also implement the Bayesian model with majority voting. We do this via tempering the likelihood of each component with temperature $t = 0.01$, i.e. $L(x) := p(x \mid \theta)^{1/t}$. This allows the posterior to concentrate on the most popular next tokens faster.

Observation Noise in the Transformer Experiments (Fig. 5, Fig. 10). To match the augmented Bayesian models, Transformers were trained on train data generated using the augmented likelihoods both in the uninformative and informative observation noise case. This meant that previously unseen tokens were present in the data component realizations. Hence the test prompts were no longer zero-probability under the training data distribution. All other experimental configurations were left unchanged.

Ablations.

- **Knockout experiment (Fig. 6):** We performed this experiment for the task of completing Dyck-3 after pre-training on three Dyck-2 language components (involving all possible bracket types. We used the best-performing model checkpoint (in terms of OOD accuracy on the nested language). As in the coinflip experiments, we replaced the outputs of each individual attention head on the test prompts with random $N(0, 1)$ noise, leaving all the other parameters unchanged. We computed the nested grammatical accuracy of the completed prompts with respect to all three train data components.
- **10 component ablation:** In order to encourage Transformers learn the general pattern of Dyck 2 languages independently from the used tokens, we carried out an ablation increasing the number of Dyck 2 components in the training data. We created two more bracket types, and trained Transformers on all possible Dyck-2 languages using two of the five bracket types. We increased

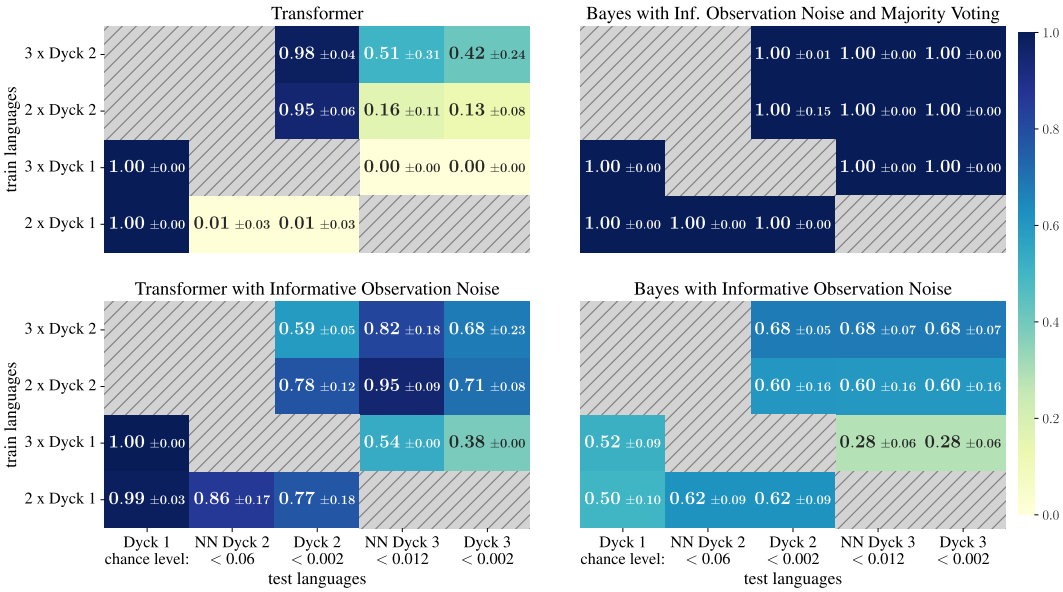

Figure 10: **Comparing Transformers to Bayes with informative observation noise on Dyck languages:** We train Transformers on simpler Dyck languages ($y-$axis, then test whether the models can finish test prompts for more complex (OOD) languages. Including more structure in the training language components (i.e., generalizing from Dyck-2 to Dyck-3) is easier for the Transformer than generalizing from simpler Dyck-1 language components. Means and standard deviations were computed from at least 4 seeds filered from 10 based on ID test accuracy.

the train data size so that the model sees the same number of examples from each component as in the 3 train component setup. We also increased the maximum number of epochs from $50,000$ to $100,000$. We generated OOD test prompts from only one Dyck-3 language - the onle containing the same tokens as the other experiments.

Additional Results. The generalized Bayes model using informative observation noise outperformed the Transformers both with and without majority voting (Fig. 10). The Transformers' performance increased when trained on data cointaining the same observation noise. These results indicate that Transfomers might implicitly implement informative observation noise better than random noise (as suggested by the uninformative noise and knockout experiments), but thie shared information content across rules is much less than in our artificial Bayesian models.

In the 10 component ablation, models achieved mean OOD accuracy of 0.181, with standard deviation 0.123, which is below than 0.42 ± 0.24 which was achieved when using three components. This indicates that seeing more examples from the same pattern (a Dyck 2 language) did not help the Transformers to synthetize the individual components into a general algorithm.

D.3 Models

We use a Transformer decoder (Vaswani et al., 2017) in flavor of the decoder-only GPT models (Radford & Narasimhan, 2018; Radford et al., 2018; et al., 2024). We apply standard positional encoding, layer normalization, ReLU activations, the AdamW optimizer (Loshchilov & Hutter, 2019) with inverse square root learning rate schedule (Xiong et al., 2020). 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.

Table 5: 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	40000

Table 6: 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	$50,000$

E ACRONYMS

CE cross entropy

AR LM autoregressive language model

IBI implicit Bayesian inference
ICL in-context learning

LLM Large Language Model
LM language model

OOD out-of-distribution