THOUGHT WITHOUT SYSTEMATICITY? EVALUATING REASONING MODELS ON RULE INDUCTION TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

A central tenet of human cognition is systematicity, the principle that understanding one concept is inherently tied to understanding close variations of that concept. Do reasoning models robustly exhibit such systematicity? If so, we would expect consistent performance on structurally equivalent variants of the same task. Here, we extend established rule induction tasks from cognitive science to assess the systematicity of thought in current reasoning models. Each task family has compositional structure that we use to create structurally equivalent task variations through task isomorphisms such as recombination and substitution. We find that despite being able to correctly solve a task, models often fail on structurally equivalent variants of the same task. These findings suggest that many model behaviors lack systematicity, rendering it difficult to robustly establish the cognitive abilities of reasoning models beyond the particular contexts they were evaluated in.

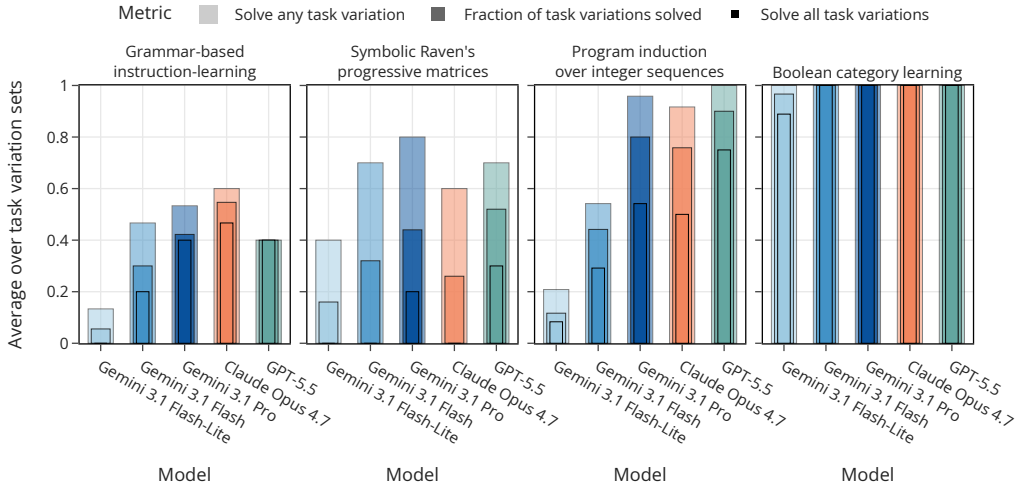


Figure 1: **Systematicity of reasoning models on rule induction tasks.** We evaluate reasoning models on compositional families of rule induction tasks. For each particular task within a family, we generate $K = 5$ structurally equivalent task variants per type of task invariance. For each set of task variations, we report whether any of the variations was solved correctly, the fraction of variations solved correctly and whether all variations were solved correctly and report the average over task variation sets within a task family. We find a notable systematicity gap between the ability to solve any task variation and solving all task variations on all task families but Boolean category learning.

1 INTRODUCTION

Systematicity is a defining feature of human language and thought: When we can understand one concept it implies that we can also understand many closely related concepts (Fodor & Pylyshyn, 1988; McLaughlin, 1993). For instance, when we learn how to *blicket*, we can also *blicket* twice, and if we are able to learn that all red squares are *wudsy* we are equally able to learn that all blue circles are *wudsy*. This systematicity reflects humans' flexibility to effortlessly generate and comprehend unseen variations of familiar elements (Chomsky, 1985; Lake et al., 2017) and it is a crucial assumption underlying cognitive tests, because it enables measuring an ability in a particular context but make inferences about that ability beyond the specific context within which it was measured.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

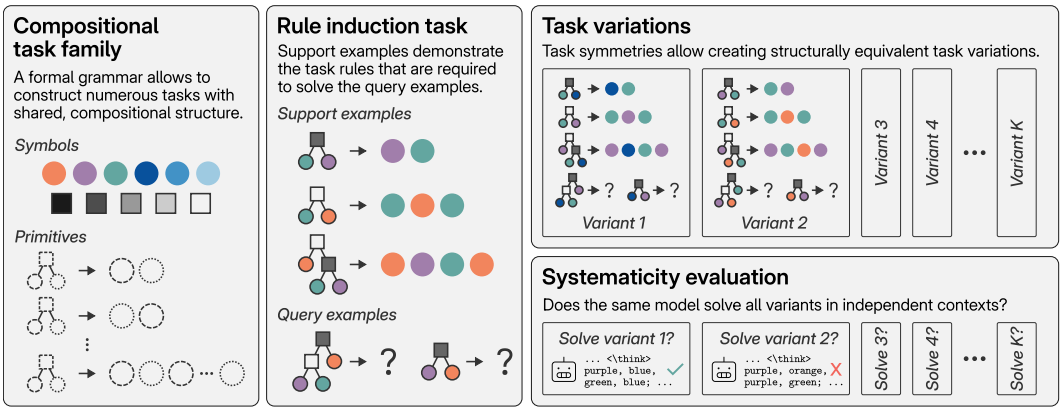


Figure 2: **Evaluating systematicity in rule induction tasks with compositional structure.** (A) Compositional task families allow to construct large numbers of tasks with related structure. The shown task is a simplified version of the grammar-based instruction-learning task presented in Section 4.1. (B) Each rule induction task requires inferring hidden rules on a set of input→output support examples to solve novel query examples. (C) The compositional task structure allows creating systematic variations of structurally equivalent tasks. (D) Reasoning models can be independently evaluated across structurally equivalent task variants to assess their systematicity.

The systematicity of human cognition has many facets reflecting the various ways in which a task can be altered according to its underlying compositional structure (Hupkes et al., 2020). This includes substitutions such as using a different symbol for the same underlying concept, or recombinations of the parts of an expression that serve similar roles. These invariances reflect compositional task structure relating tasks to each other through a common syntax that prescribes how sets of primitives can be composed into different configurations. From this perspective, systematicity can be understood as a behavioral outcome of successfully capturing the underlying compositional structure of a family of tasks. Compositional task families are exponential in nature and it is virtually impossible to exhaustively include all possible task variations in the training data even at large scale (Schug, 2025). This raises the question: *Do reasoning models exhibit systematicity of thought and perform consistently across the many possible, structurally equivalent variations of a given task?*

Here, we study this question by drawing on established experimental paradigms from cognitive science. In particular, we consider rule induction tasks that require the learner to infer latent structure, and evaluate it across structurally equivalent task variations to probe the systematicity of its thinking as illustrated in Figure 2. A variety of such tasks have been studied in cognitive science, ranging from simple Boolean concept learning, abstract reasoning over symbolic sequences, to discovering the syntax and semantics of artificial languages (Raven, 1962; Feldman, 2000; Piantadosi et al., 2016; Lake & Baroni, 2023; Rule et al., 2024). These tasks have compositional structure that allows us to generate a large number of possible tasks and probe understanding beyond the specifics of any particular experience. Importantly, for each specific task we can create structurally equivalent task variations by using *task isomorphisms* – equivariant task transformations based on the underlying compositional structure – to evaluate systematicity. Unlike humans whose mental states are invariably altered by experience, models provide the opportunity to evaluate systematicity in its strictest form by observing behavioral outcomes in multiple counterfactual worlds that do not affect each other.

We summarize our **main contributions** in the following:

- In Section 3, we formalize systematicity and how to measure it over structurally equivalent rule induction tasks via task isomorphisms.
- In Section 4 and Appendix B.1, we adapt four rule induction task families for generating systematic task variations: Grammar-based instruction learning (Lake & Baroni, 2023), a symbolic version of Raven's progressive matrices (Schug et al., 2025), program induction over integer sequences (Rule et al., 2024) and Boolean category learning (Piantadosi et al., 2016).
- In Section 5, we evaluate the systematicity of reasoning models across task variations on our four task families and investigate the impact of sampling noise. In Appendix C.1, we additionally examine how reasoning effort impacts systematicity.

2 Related work

The observation that human cognition is systematic has been a longstanding locus of inquiry, reaching back to early attempts to formulate laws of thought (Boole, 1854; Griffiths, 2026). Systematicity is a central tenet of the language of thought hypothesis – a prominent theory of human cognition which assumes that mental representations have combinatorial syntax and semantics (Fodor, 1975; Fodor & Pylyshyn, 1988; Penn et al., 2008; Goodman et al., 2015). Cognitive science therefore has a rich history of developing tests of rule learning, with varying latent syntactic and semantic structure, to investigate the language of thought hypothesis. Classic work by Feldman (2000) uses Boolean concept learning tasks to show that subjective difficulty is directly related to the complexity of the underlying rule and learning tasks with more complex logical rules have been used to investigate what concrete primitives might underlie the language of thought (Piantadosi et al., 2016; Rule et al., 2024). Tasks that require inferring compositional string rewriting rules have further been used to compare compositional learning between humans and neural networks, a challenge neural networks have historically struggled with but recently made notable progress on (Lake & Baroni, 2023).

Despite this progress, early large language models were brittle and could be derailed by subtle variations in their inputs, including typos, irrelevant sentences, changes in phrasing, and reordering of examples (Jia & Liang, 2017; Jiang et al., 2020; Zhao et al., 2021; Lu et al., 2022). While instruction tuning via human feedback (Ouyang et al., 2022), progressive scaling (Kaplan et al., 2020; Bubeck et al., 2023) and inference-time reasoning (Wei et al., 2022; Wang et al., 2022; Lightman et al., 2024; OpenAI et al., 2024) have rendered the resulting models significantly less sensitive to surface changes, adversarial brittleness remains a central concern (Wang et al., 2023; Zou et al., 2023; Berglund et al., 2024; Romanou et al., 2026; Mondorf et al., 2026; Burnell et al., 2026), in particular in the context of safety and alignment (Wei et al., 2023; Chen et al., 2025).

In recent years, stronger emphasis has therefore been put on evaluating model consistency through metamorphic testing where a known relationship between outputs of related inputs is evaluated (Segura et al., 2016). This includes consistency across languages (Cho et al., 2025) or category-based substitution (Ribeiro et al., 2020). Logical consistency testing in particular evaluates the internal consistency of model knowledge (Jang et al., 2022), e.g. by verifying invariance to paraphrasing (Elazar et al., 2021) or whether predicted relations are closed under transitivity (Jang & Lukasiewicz, 2023). Our systematicity evaluation can be interpreted as a specific type of metamorphic test that verifies invariant task-solving ability with respect to a compositional data generating procedure.

3 What is systematicity and how can we measure it?

In the following, we formalize systematicity, how it can be measured in the setting of rule induction tasks and develop metrics to quantify a learner's systematicity on such tasks.

3.1 What is systematicity?

We define systematicity as invariance of ability with respect to certain transformations of a task with compositional structure. For example, we might expect the ability to understand an expression like "Alice loves John" to be invariant to a permutation of its constituents like "John loves Alice". This means we treat systematicity as a latent property of behavior that makes assumptions about the compositional structure underlying a data generating process. To quantify systematicity, we will measure how the ability to solve a task varies across task variations that modify the constituents of a task according to its compositional structure. Since some task variations might change the difficulty of a task and confound the systematicity measure, we will restrict task variations to be structurally equivalent to each other through isomorphisms, one-to-one mappings between each task variation.

3.2 Types of systematic task variations

The particular ways in which task constituents can be modified depend on the compositional structure of the data generating process. In our task families we will consider the following task transformations:

- **Symbol substitution**: Replacing symbols that carry no intrinsic semantic meaning relevant to the task. For example replacing one pseudoword "dax" with another pseudoword "lug".

- **Feature rebinding**: Rebinding latent variables to different features. For example, representing the same number through size, orientation or numerosity.
- **Constituent permutation**: Shuffling constituents of the same type within an expression. For example, "Alice loves John" $\leftrightarrow$ "John loves Alice".
- **Example reordering**: Changing the order with which multiple independent examples are presented.

3.3 Structurally equivalent rule induction tasks

We will evaluate systematicity within the setting of rule induction tasks as illustrated in Figure 2. In each task, we present a learner with a set of input-output examples based on which it has to infer hidden underlying rules. We then verify whether the learner correctly inferred the rules by asking it to predict the outputs on a novel set of inputs. For instance, in the Boolean category learning task, a learner might be presented with three support examples, ■ $\rightarrow$ `wudsy`, ■ $\rightarrow$ `not wudsy`, ▲ $\rightarrow$ `wudsy`, from which it has to infer the simplest rule that explains which objects are `wudsy` (*all red objects are wudsy*). We then evaluate whether it did so correctly by asking it to complete query examples such as, ■ $\rightarrow$ **?** and ● $\rightarrow$ **?**.

To study a learner's systematicity across variations of a task of equal difficulty, we construct structurally equivalent tasks through *isomorphisms*. An isomorphism is a structure-preserving mapping that is invertible and can therefore only alter the surface characteristics of a task but not its underlying structure. For example, we can construct the structurally equivalent task ● $\rightarrow$ `wudsy`, ● $\rightarrow$ `not wudsy`, ■ $\rightarrow$ `wudsy`, ● $\rightarrow$ **?** and ▲ $\rightarrow$ **?**, where the hidden rule is *all blue objects are wudsy* and the colors and shapes were remapped accordingly.

3.4 Isomorphic task variations

Formally, let $\mathcal{X}$ and $\mathcal{Y}$ be input and output spaces. We define a task as a tuple $T = (f, S, Q)$, where $f : \mathcal{X} \rightarrow \mathcal{Y}$ is a *hidden rule*, $S = ((x_i, f(x_i)))_{i=1}^{N}$ is the *support set*, and $Q = (x'_j)_{j=1}^{M}$ is the *query set*, with $x_i, x'_j \in \mathcal{X}$. Upon observing only the support set S, a learner is tasked to predict the targets of the query set, $f(x'_j)$ for all $j = 1, \ldots, M$. We say a learner solves a task if it correctly predicts the whole query set.

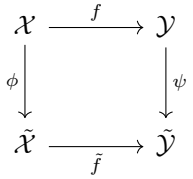


Figure 3: Relationship between the original and transformed hidden rule.

An *isomorphic task variation* is generated by applying a transformation $\tau = (\pi, \sigma, \phi, \psi)$, where π and σ are permutations of the indices $\{1, \ldots, N\}$ and $\{1, \ldots, M\}$ respectively, and $\phi : \mathcal{X} \rightarrow \tilde{\mathcal{X}}$ and $\psi : \mathcal{Y} \rightarrow \tilde{\mathcal{Y}}$ are bijections that map elements from the original input and output space to new input and output spaces $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{Y}}$. Applying τ yields a transformed task $\tau(T) = (\tilde{f}, \tilde{S}, \tilde{Q})$, where the new hidden rule evaluates as $\tilde{f} = \psi \circ f \circ \phi^{-1}$, the transformed and permuted support set is $\tilde{S} = \left(\phi(x_{\pi(i)}), \psi(f(x_{\pi(i)}))\right)_{i=1}^{N}$, and the transformed and permuted query set is $\tilde{Q} = \left(\phi(x'_{\sigma(j)})\right)_{j=1}^{M}$.

3.5 Systematicity metrics

A strictly systematic learner should be invariant to isomorphic task variations assuming that the semantics of any two input spaces are equally (un)informative for solving the task: If it can grasp the underlying structure well enough to solve one task variation, we would expect it to be able to solve them all. Let $\text{pass}(T) \in \{0, 1\}$ be a binary indicator denoting whether a learner correctly predicts the targets for the entire query set of task T. To evaluate a learner's robustness to isomorphic variations, we sample K task transformations $\{\tau_1, \ldots, \tau_K\}, \tau_k \sim p(\tau)$ for each task T and calculate the following systematicity metrics:

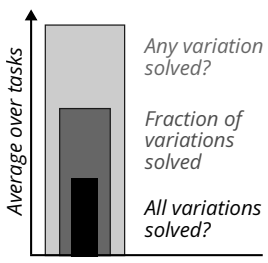


Figure 4: We use bullet charts to compactly visualize the three systematicity metrics.

- **Solve any of K variations**: $\max_{k \in \{1,\ldots,K\}} \text{pass}(\tau_k(T))$
- **Fraction of variations solved**: $\frac{1}{K} \sum_{k=1}^{K} \text{pass}(\tau_k(T))$
- **Solve all K variations**: $\min_{k \in \{1,\ldots,K\}} \text{pass}(\tau_k(T))$

Collectively, these three metrics allow us to characterize the systematicity with which a learner solves a task: If it never passes any task variation, this means the tasks are too difficult for the learner. If it passes some task variations but not all of them, it is not fully systematic and the fraction of variations passed captures to what extent. If all task variations are passed it can be considered systematic on this task with respect to the tested task variations. We can take advantage of the fact that the three metrics are monotonically decreasing and visualize them as a bullet chart as shown in Figure 4.

3.6 Systematicity under stochasticity

An important consideration in the context of systematicity is how to handle possible stochasticity in the learner that solves the tasks. When repeatedly evaluating a stochastic learner on a given task, the learner might randomly fail to complete the task on some attempts. While large language models were traditionally evaluated using greedy decoding, current reasoning models have been found to perform better when evaluated with stochastic sampling (Wang et al., 2022; Guo et al., 2025). In fact, many proprietary model providers now only allow to use reasoning models with a positive sampling temperature. To account for the resulting decoding variance, we give each reasoning model multiple attempts per task variation (here we use $N = 5$ attempts throughout) and consider a variation as passed if it was correctly solved in the majority of the attempts. We will further study the impact of stochasticity on systematicity in Section 5.2.

4 Rule induction tasks for evaluating systematicity

In the following sections, we present four families of rule induction tasks built on established task paradigms from cognitive science. Each data generating procedure relies on some form of compositional structure that allows to sample a large number of compositional tasks of varying difficulty. Their synthetic nature provides us with the necessary control to create task variations that are guaranteed to be isomorphic in order to evaluate systematicity. Please see Appendix B.1 for the description of the Boolean category learning task.

4.1 Grammar-based instruction-learning

The grammar-based instruction-learning task family was introduced by Lake & Baroni (2023). In each task from this family, the model must infer the latent grammatical rules of an artificial language from a few demonstrations to translate pseudolanguage commands into a sequence of outputs. Since the artificial languages are procedurally generated from a meta-grammar, an infinite number of such languages of varying complexity can in principle be generated. This task family was originally designed to evaluate the ability of humans and neural networks for compositional generalization, the ability to solve unseen task compositions made from familiar parts.

Instructions To reduce the influence of prior experience on task performance as well as limit possible ambiguity, we provide detailed instructions on the general structure of each task in the system prompt, shown in Figure 11.

Example The following is a sample episode from the grammar-based instruction-learning task family. To predict the single query example of this task, the learner must infer primitive rules, mapping input tokens, like `dax`, `lug`, to output tokens, `RED`, `BLUE`, and function rules that transform their inputs, like `fep`, `zup`. Here `fep` triples its input and `zup` swaps its inputs. The learner then has to apply the inferred rules to a novel input composition and predict its answer, here 🔴🔴🔴🔵🔵🔵.

`dax fep` → 🔴🔴🔴
`lug fep` → 🔵🔵🔵
`dax zup lug` → 🔵🔴
`dax zup lug fep` → 🔵🔵🔵🔴
`dax fep zup lug` → 🔵🔴🔴🔴
`lug fep zup dax fep` → **?**

Task structure Each generated language is governed by a uniquely sampled syntax that consists of primitive rules and function rules. Whereas primitive rules are simple one-to-one mappings between input tokens (e.g., pseudowords like `dax`, `lug`) and output tokens (e.g., capitalized colors like `RED`, `BLUE`), function rules specify how function tokens transform one or two adjacent arguments into a sequence of outputs. The function arguments are either a single primitive token or match a whole preceding or succeeding string. When applied, function rules deterministically reorder, delete or duplicate their arguments to produce a new output

sequence. To resolve syntactic ambiguities, functions that accept string arguments have a strict precedence order. In addition, the parser evaluates sequences via left-to-right reduction, ensuring that for any valid input string there exists a unique translation.

Support and query set In the original grammar-based instruction-learning task family as used in Lake & Baroni (2023), tasks were hand selected to ensure that their underlying grammar could be unambiguously inferred. Since we would like to generate a large number of unambiguous tasks, we define a procedure to automatically create an instructive support set that ensures all primitive rules and function rules can be identified for a given, randomly sampled grammar . Specifically, we construct the support set by producing lexical anchors that demonstrate the primitive rules (e.g., `dax fep` → 🔴🔴🔴 and `lug fep` → 🔵🔵🔵 in the example above), template resolvers that reveal the arity and output transformations of the rule functions (e.g., `dax zup lug` → 🔵🔴 shows that `zup` is a two argument reversal function), and precedence proofs that disambiguate the precedence order of multiple string matching function rules. The query set as well as a configurable number of additional support examples is then generated from nested compositions that involve at least two function rule applications per example, ensuring that there are no duplicate examples across the support and query set.

Task invariances The resulting tasks have several invariances that we can use to generate isomorphic task variations. We can permute the order of both the support and query examples (*example reordering*), apply a bijective mapping to the input and output vocabularies (*symbol substitution*) and permute the primitive tokens in a given expression, e.g. by changing `dax zup lug` to `lug zup dax` (*recomposition*).

4.2 Symbolic Raven's progressive matrices

Raven's progressive matrices is a classic human intelligence test (Raven, 1962). In its original form, each task consists of a three by three grid of abstract symbols with a missing final panel whose contents need to be inferred. Schug et al. (2025) introduce a symbolic variant of this task family that allows to procedurally generate Raven-like tasks of varying difficulty, creating challenging abstract reasoning problems.

Instructions Similar to before we provide detailed instructions on the general structure of tasks from this task family in the system prompt, shown in Figure 12.

Example The following is a sample episode from the symbolic Raven's progressive matrices task family using two ($F = 2$) and modulo 10 arithmetic. The left side shows the unpermuted base task, while the right side additionally considers a column-specific feature permutation which models the difficulty of *finding correspondences* (Carpenter et al., 1990). To solve the unpermuted task (left), the learner must infer the rules applied horizontally to each aligned feature dimension. Here, the first feature is constant (`8` → `8` → `8`), and the second feature follows an arithmetic progression of +2 modulo 10 (e.g., `7` → `9` → `1`). Applying these rules to the third row we obtain the target panel, $\begin{bmatrix}3\\9\end{bmatrix}$.

$$\begin{bmatrix}5\\2\end{bmatrix}\begin{bmatrix}5\\4\end{bmatrix}\begin{bmatrix}5\\6\end{bmatrix}\begin{bmatrix}8\\7\end{bmatrix}\begin{bmatrix}8\\9\end{bmatrix}\begin{bmatrix}8\\1\end{bmatrix}\begin{bmatrix}3\\5\end{bmatrix}\begin{bmatrix}3\\7\end{bmatrix}\begin{bmatrix}\mathbf{?}\\\mathbf{?}\end{bmatrix} \Leftrightarrow \begin{bmatrix}2\\5\end{bmatrix}\begin{bmatrix}5\\4\end{bmatrix}\begin{bmatrix}6\\5\end{bmatrix}\begin{bmatrix}7\\8\end{bmatrix}\begin{bmatrix}8\\9\end{bmatrix}\begin{bmatrix}1\\8\end{bmatrix}\begin{bmatrix}5\\3\end{bmatrix}\begin{bmatrix}3\\7\end{bmatrix}\begin{bmatrix}\mathbf{?}\\\mathbf{?}\end{bmatrix}$$

In the permuted variant (right), the task is significantly more difficult because the features are no longer spatially aligned across columns. The learner must simultaneously discover the latent rules and the implicit feature correspondence – recognizing, for example, that the constant feature corresponds to the bottom element of column 1, the top element of column 2, and the bottom element of column 3. After disentangling these mappings and applying the latent rules, the learner must predict the appropriately permuted final panel, $\begin{bmatrix}9\\3\end{bmatrix}$.

Task structure Each task is structured as a 3×3 grid of panels, where each panel contains an F-dimensional feature vector of integers, governed by a hidden combination of F rules and ordered according to a column-specific permutation. The rules operate horizontally, such that the features of the third column in each row are determined by applying independent rules to the features in the first two columns. The same sequence of rules is applied consistently across all three rows. The

available rules encompass constant patterns, arithmetic progressions, modular addition or subtraction, minimum/maximum operations, and the distribution of distinct elements.

Support and query set To generate a task, we sample a hidden combination of rules, one for each feature dimension as well as column-specific feature permutations. The first two rows of the 3×3 matrix act as the support set, demonstrating the applied rules, while the first two columns of the third row serve as the query.

Task invariances The column-specific feature permutations can be used to create isomorphic transformations of the same task, rebinding the hidden features of the unpermuted task to the observed feature orderings (*feature rebinding*).

4.3 Program induction over integer sequences

Next, we consider rule induction over integer sequences as studied by Rule et al. (2024) in which a learner must infer a latent rule to transform an input list of integers into an output list of integers. Since the original set of tasks were handcrafted, we define a probabilistic context-free grammar that allows us to procedurally sample a large number of similar tasks and systematically create isomorphic task variations.

Instructions We provide the general task description shown at the top of Figure 13 to the learner in the main evaluation and perform additional experiments where we additionally describe the basic operations from which each task is composed in the instructions as shown at the bottom of Figure 13.

Example The following is a sample episode from the program induction over integer sequences task family. To predict the query in this example, the learner must infer that the target rule is a composition of two operations: first swapping the two halves of the list, and then removing the element at the first position. It then has to apply these operations to the novel test input sequence to predict the answer (here the answer is `[5, 2, 7, 3]`).

```
[1, 2, 3, 4] → [4, 1, 2]
   [1, 8, 3] → [3, 1]
[8, 9, 1, 2, 4] → [2, 4, 8, 9]
      [0, 1] → [0]
[3, 3, 4, 5, 5, 6] → [5, 6, 3, 3, 4]
         [2] → []
[7, 3, 1, 5, 2] → ?
```

Task structure The underlying program for each task is sampled from a probabilistic context-free grammar that allows to create compositions of operations that take integer lists as input and output integer lists. The grammar supports parameterized operations such as `insert` or `remove`, which take additional integer arguments such as structural positions, sequence lengths, or vocabulary elements. The full list of operations is shown at the bottom of Figure 13. By composing these operations, the generative model produces a single deterministic expression that parses and transforms any given input list. Parameters and input sequence integers are sampled uniformly from predefined ranges.

Support and query set We randomly sample input lists of integers in the range of zero to nine for the support and query set, strictly partitioning the sequence lengths of the support and query sets into disjoint pools. Specifically, we ensure that the support set contains lists that are both shorter and longer than the lists in the query set. The query set examples are subsequently sampled from intermediate sequence lengths that are never seen in the support set. This is to prevent the existence of a shortcut solution in which – rather than inferring the underlying program – it would be possible to simply copy the analogical elements of the same length example.

Task invariances We can create isomorphic task variations by permuting the order of the support and query samples (*example reordering*). In addition, we can apply a bijective mapping to the integer vocabulary of the elements in the support and query set as well as implicitly to any integer parameters in the hidden rule, e.g., the value argument inserted by the `insert` operation (*symbol substitution*).

5 Evaluating reasoning models across systematic task variations

We now evaluate systematicity of current state-of-the-art proprietary reasoning models on the rule induction task families introduced above. For each task, we generate a set of isomorphic task variations, compute the systematicity metrics defined in Section 3.5 and report the average across task variation sets within a task family. While ideally we would cover all possible variations, there

is typically a prohibitively large number of possible variations. Throughout our experiments, we therefore sample $K = 5$ variations per task isomorphism.

Apart from experiments where we explicitly vary the reasoning effort (see Appendix C.1), we evaluate all models with reasoning effort set to 'high' and use the default sampling parameters of the official APIs for each model. We use structured outputs to ensure models output consistently formatted responses according to the schema requirements of each task family. We compare models from three major model providers in the main evaluation and focus on Gemini models to examine the effect of reasoning effort to keep costs reasonable.

5.1 Models can solve some task variations but struggle to systematically solve all variations

Figure 1 shows average systematicity metrics over task variation sets on the four rule induction task families. Across most of the task families, we find that there is a notable systematicity gap, i.e. a clear difference between the fraction of task variation sets where any variation was solved and the fraction of task variation sets where all variations were solved. The *Boolean category learning* task is a notable exception to this observation. Almost all models can solve all tasks and variations thereof, demonstrating that perfect systematicity is achievable.

The systematicity gap appears most clearly for *symbolic Raven's progressive matrices* and *program induction over integer sequences*. For example, in 91.67% of the task variation sets in the program induction task family, Claude Opus 4.7 can solve at least one variation but only in 50.00% of the task variation sets can it solve all variations, leaving a systematicity gap of 41.67%.

We can decrease the difficulty of the program induction tasks by providing a detailed *hint* of the possible program components (see Figure 13) and accordingly observe a notable improvement in performance for stronger models in Figure 9 albeit without fully resolving the systematicity gap. The Raven's matrices task family already uses the easiest parameterization with only two features. Making it more difficult by increasing the number of features to three predictably further decreases model performance further, as shown in Figure 10.

On the *grammar-based instruction-learning* task family overall performance is modest and models lack systematicity with the notable exception of GPT-5.5: if it can solve any task variation it can solve all of them, otherwise it solves none. Figure 8A further demonstrates that the systematicity gap is not specific to a particular type of task isomorphism but appears to varying degrees for recomposition, reordering and substitution. In addition, Figure 8B shows that while adding extra support examples alongside the curriculum examples makes the task easier, the systematicity gap remains.

5.2 Model stochasticity only partially explains the systematicity gap

Since reasoning models rely on stochastic sampling, two sources of variance affect model systematicity: the within-task-variation sampling noise and the between-task-variation variance due to the isomorphic task variations. In the following, we try to delineate the impact of either, first by decomposing the empirically observed variances and then by re-evaluating models with zero temperature sampling to eliminate sampling noise (up to numerical stability).

Large variance for multiple attempts on the same task. By applying the law of total variance and using suitable empirical estimators, we can decompose the total variance across task variations and attempts for a given task into the within-task-variation variance and the between-task-variation variance. In Appendix A, we formally state this decomposition and develop the respective estimators. Table 1 lists the different variance components across task families, type of task variation and model. We find that within-task-variation variance often makes up the largest part of the total variance implying that large sampling noise dominates (or rather masks as we will see in the next section) between-task-variation variance.

Systematicity gap persists with greedy decoding By setting the sampling temperature to zero we can, in principle, eliminate the within-task-variation variance. We re-evaluate Gemini models[1] in

[1]Neither GPT-5.5 nor Claude Opus 4.7 support zero temperature reasoning, so we have to limit this evaluation to Gemini.

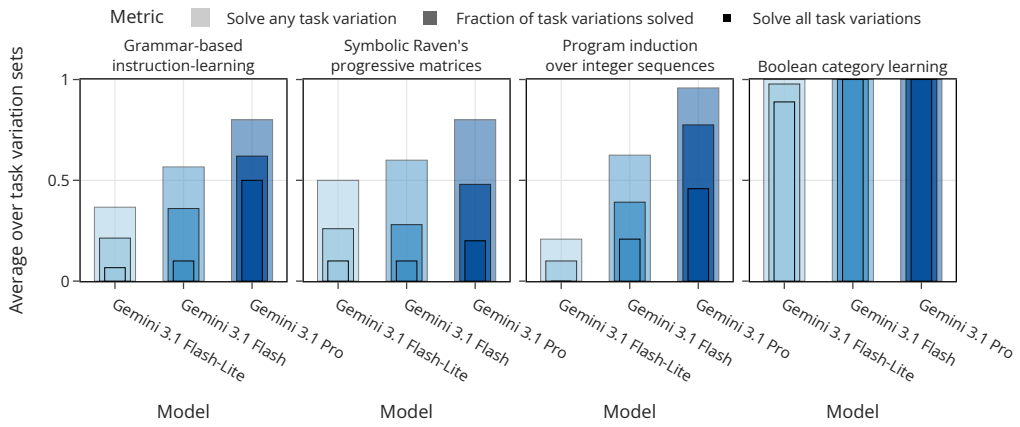


Figure 5: **Systematicity gap persists with greedy decoding.** When setting the decoding sampling temperature to zero during inference, reasoning model task performance remains comparable and the systematicity gap persists.

this quasi-deterministic setting[2], finding that the systematicity gap and model performance remains largely unchanged, as shown in Figure 5. This suggests that the lack of systematicity observed in reasoning models goes beyond the noisy nature of stochastic decoding.

6 Discussion

We investigated systematicity of thought of reasoning models on four rule induction task families – grammar-based instruction-learning, symbolic reasoning, program induction over integer sequences and Boolean concept learning. Based on the compositional structure of each task family, we identify task isomorphisms that allow us to generate structurally equivalent task variations. Across most task families we find that reasoning models lack strict systematicity of thought: despite being able to solve a task, they often fail to solve structurally equivalent variants of it. These results demonstrate the difficulty of robustly establishing the cognitive abilities of reasoning models beyond the particular contexts they are evaluated in. Our findings further raise the following points of discussion:

Reliability is a prerequisite for systematicity. Consistent behavior when repeatedly facing the exact same context is a requirement for systematicity. Our findings reveal that even with majority voting over five independent attempts, current reasoning models often do not satisfy this requirement.

Machines should strive to be more systematic than humans. We can evaluate systematicity in reasoning models in its strictest form by comparing independent behavioral outcomes across task variations given the exact same initial conditions. This is not possible with humans where any experience will alter future behavioral outcomes. Nevertheless, humans are arguably not perfectly systematic either: Many variables such as mood, tiredness or semantics can affect performance despite having the competence to solve a task in principle (Chomsky, 1965; Wason, 1968; Evans et al., 1983; Ashby et al., 1999; Van Der Linden et al., 2003). In comparison, the intelligent machines we build are tireless and we should strive for them to be as systematic as possible.

Test validity requires systematicity. A lack of systematicity raises a fundamental issue of test validity in the evaluation of reasoning models: Tests have to commit to a particular context and therefore differ by definition from the plethora of contexts within which a system is expected to operate. Without systematicity, we lack a strong reason to expect that the ability to do something in one context is predictive of this ability in other contexts.

Limitations Our evaluation relies on synthetic data generation in order to create the controlled, isomorphic task variations needed to isolate systematicity. Since measuring systematicity requires evaluating many reasoning chains per task, it is costly and we limit the number of task variations we generate per task to $K = 5$ (each of which is evaluated over $N = 5$ attempts to compute the majority vote). Ultimately, a strictly systematic system should solve all task variations, regardless of whether it is evaluated on $K = 5$ or $K = 100$ variations.

[2]Floating point arithmetic can lead to residual stochasticity during greedy decoding.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we are providing the code to run our evaluation on any model available through an API as part of the supplementary material. The system prompts are additionally listed in Appendix D and software dependencies are detailed in Appendix E.1.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments, implement methods, and assist with translation. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, , propose or refine hypotheses, support qualitative and thematic data analysis, and interpret results. "Assist in the writing of proofs, provide critical ingredients for proving mathematical claims, and clean and reformat dataset" are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, suggest experimental parameters, create or edit software code, summarize or analyse existing literature, discover research topics or identify gaps, brainstorming, sourcing/searching for information, edit a research paper to improve readability, and identify relevant literature. We have reviewed all AI-assisted work: All code is carefully tested and every single line of code written is reviewed by a human before being committed. Any literature suggested via AI is manually retrieved from the source and any claims thereof validated in the primary text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

F. Gregory Ashby, Alice M. Isen, and And U. Turken. A neuropsychological theory of positive affect and its influence on cognition. *Psychological Review*, 106(3):529–550, 1999. ISSN 1939-1471, 0033-295X. doi: 10.1037/0033-295X.106.3.529. URL https://doi.apa.org/doi/10.1037/0033-295X.106.3.529.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Stickland, Tomasz Korbak, and Owain Evans. The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A". In *I Can't Believe It's Not Better Workshop: Failure Modes in the Age of Foundation Models*, 2024. URL https://openreview.net/forum?id=TGTZwVabpU.

Lukas Biewald. Experiment Tracking with Weights and Biases, 2020. URL https://www.wandb.com/.

George Boole. *An investigation of the laws of thought,*. New York, Dover, 1854. URL http://archive.org/details/investigationofl01bool.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of Artificial General Intelligence: Early experiments with GPT-4, April 2023. URL http://arxiv.org/abs/2303.12712. arXiv:2303.12712 [cs].

Ryan Burnell, Yumeya Yamamori, Orhan Firat, Kate Olszewska, Steph Hughes-Fitt, Oran Kelly, Isaac R. Galatzer-Levy, Meredith Ringel Morris, Allan Dafoe, Alison M. Snyder, Noah D. Goodman, Matthew Botvinick, and Shane Legg. Measuring Progress Toward AGI: A Cognitive Framework, May 2026. URL http://arxiv.org/abs/2605.28405. arXiv:2605.28405 [cs.AI] version: 1.

P. A. Carpenter, M. A. Just, and P. Shell. What one intelligence test measures: a theoretical account of the processing in the Raven Progressive Matrices Test. *Psychological Review*, 97 (3):404–431, July 1990. ISSN 0033-295X. URL https://psycnet.apa.org/record/1990-27436-001.

Yueh-Han Chen, Guy Davidson, and Brenden Lake. SAGE-Eval: Evaluating LLMs for Systematic Generalizations of Safety Facts. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/75a5df8d390c51d6aebfa2accb3f4911-Paper-Datasets_and_Benchmarks_Track.pdf.

Steven Cho, Stefano Ruberto, and Valerio Terragni. Metamorphic Testing of Large Language Models for Natural Language Processing. In *2025 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 174–186, September 2025. doi: 10.1109/ICSME64153.2025.00025. URL https://ieeexplore.ieee.org/document/11185922. ISSN: 2576-3148.

Noam Chomsky. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, 1965. ISBN 978-0-262-32779-4.

Noam Chomsky. *Syntactic structures*. Number 4 in Janua Linguarum Series minor. Mouton Publ, The Hague, 14. printing edition, 1985. ISBN 978-90-279-3385-0.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and Improving Consistency in Pretrained Language Models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021. doi: 10.1162/tacl_a_00410. URL https://aclanthology.org/2021.tacl-1.60/.

J. St. B. T. Evans, Julie L. Barston, and Paul Pollard. On the conflict between logic and belief in syllogistic reasoning. *Memory & Cognition*, 11(3):295–306, May 1983. ISSN 0090-502X, 1532-5946. doi: 10.3758/BF03196976. URL https://link.springer.com/10.3758/BF03196976.

Jacob Feldman. Minimization of Boolean complexity in human concept learning. *Nature*, 407 (6804):630–633, October 2000. ISSN 1476-4687. doi: 10.1038/35036586. URL https://www.nature.com/articles/35036586.

J.A. Fodor. *The Language of Thought*. Language and thought series. Harvard University Press, 1975. ISBN 978-0-674-51030-2. URL https://books.google.com/books?id=XZwGLBYLbg4C.

Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, March 1988. ISSN 00100277. doi: 10.1016/0010-0277(88)90031-5. URL https://linkinghub.elsevier.com/retrieve/pii/0010027788900315.

Noah D. Goodman, Joshua B. Tenenbaum, and Tobias Gerstenberg. Concepts in a probabilistic language of thought. In *The Conceptual Mind: New Directions in the Study of Concepts*, pp. 623–653. MIT Press, 2015.

Tom Griffiths. *The Laws of Thought: The Quest for a Mathematical Theory of the Mind*. HarperCollins Publishers, February 2026. ISBN 978-0-00-871150-4.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081): 633–638, September 2025. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality Decomposed: How do Neural Networks Generalise? *Journal of Artificial Intelligence Research*, 67:757–795, April 2020. ISSN 1076-9757. doi: 10.1613/jair.1.11674. URL https://www.jair.org/index.php/jair/article/view/11674.

Plotly Technologies Inc. Collaborative data science, 2015. URL https://plot.ly. Place: Montreal, QC.

Myeongjun Jang and Thomas Lukasiewicz. Consistency Analysis of ChatGPT. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15970–15985, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.991. URL https://aclanthology.org/2023.emnlp-main.991/.

Myeongjun Jang, Deuk Sin Kwon, and Thomas Lukasiewicz. BECEL: Benchmark for Consistency Evaluation of Language Models. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na (eds.), *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 3680–3696, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.324/.

Robin Jia and Percy Liang. Adversarial Examples for Evaluating Reading Comprehension Systems. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1215. URL https://aclanthology.org/D17-1215/.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020. doi: 10.1162/tacl_a_00324. URL https://aclanthology.org/2020.tacl-1.28/.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. URL http://arxiv.org/abs/2001.08361. arXiv:2001.08361 [cs].

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.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, January 2017. ISSN 0140-525X, 1469-1825. doi: 10.1017/S0140525X16001837. URL https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/building-machines-that-learn-and-think-like-people/A9535B1D745A0377E16C590E14B94993.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let' s Verify Step by Step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556/.

Charlie Marsh. uv: An extremely fast Python package and project manager, written in Rust., 2024. URL https://pypi.org/project/uv/.

Brian P. McLaughlin. The connectionism/classicism battle to win souls. *Philosophical Studies*, 71(2):163–190, August 1993. ISSN 1573-0883. doi: 10.1007/BF00989855. URL https://doi.org/10.1007/BF00989855.

Philipp Mondorf, Samuel J. Bell, Jesse Dodge, and Dieuwke Hupkes. LPDS: Evaluating LLM Robustness Through Logic-Preserving Difficulty Scaling, 2026. URL https://arxiv.org/abs/2605.15393. _eprint: 2605.15393.

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Bohan Zhang, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wenting Zhan, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. OpenAI o1 System Card. Technical report, 2024. URL https://arxiv.org/abs/2412.16720. _eprint: 2412.16720.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll 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 F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Ad-*

vances in Neural Information Processing Systems, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Derek C. Penn, Keith J. Holyoak, and Daniel J. Povinelli. Darwin's mistake: Explaining the discontinuity between human and nonhuman minds. *Behavioral and Brain Sciences*, 31(2): 109–130, 2008. ISSN 1469-1825. doi: 10.1017/S0140525X08003543.

Steven T. Piantadosi, Joshua B. Tenenbaum, and Noah D. Goodman. The logical primitives of thought: Empirical foundations for compositional cognitive models. *Psychological Review*, 123(4):392–424, July 2016. ISSN 1939-1471, 0033-295X. doi: 10.1037/a0039980. URL https://doi.apa.org/doi/10.1037/a0039980.

John C. Raven. *Advanced Progressive Matrices, Set II*. H. K. Lewis, London, 1962.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.442. URL https://aclanthology.org/2020.acl-main.442/.

Angelika Romanou, Mark Ibrahim, Candace Ross, Chantal Shaib, Kerem Oktar, Samuel J. Bell, Anaelia Ovalle, Jesse Dodge, Antoine Bosselut, Koustuv Sinha, and Adina Williams. Brittlebench: Quantifying LLM robustness via prompt sensitivity, 2026. URL https://arxiv.org/abs/2603.13285. _eprint: 2603.13285.

Joshua S. Rule, Steven T. Piantadosi, Andrew Cropper, Kevin Ellis, Maxwell Nye, and Joshua B. Tenenbaum. Symbolic metaprogram search improves learning efficiency and explains rule learning in humans. *Nature Communications*, 15(1):6847, August 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-50966-x. URL https://www.nature.com/articles/s41467-024-50966-x.

Simon Schug. *Meta-Learning & Compositional Generalization in Neural Networks*. PhD thesis, ETH Zurich, 2025. URL http://hdl.handle.net/20.500.11850/738789.

Simon Schug, Seijin Kobayashi, Yassir Akram, Joao Sacramento, and Razvan Pascanu. Attention as a Hypernetwork. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 68744–68770, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/abfa542f546df3c6c35695ec8d5bf4b9-Paper-Conference.pdf.

Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-Cortés. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering*, 42(9):805–824, September 2016. ISSN 1939-3520. doi: 10.1109/TSE.2016.2532875. URL https://ieeexplore.ieee.org/document/7422146.

Dimitri Van Der Linden, Michael Frese, and Theo F Meijman. Mental fatigue and the control of cognitive processes: effects on perseveration and planning. *Acta Psychologica*, 113(1):45–65, May 2003. ISSN 00016918. doi: 10.1016/S0001-6918(02)00150-6. URL https://linkinghub.elsevier.com/retrieve/pii/S0001691802001506.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, Sang Truong, Simran Arora, Mantas Mazeika, Dan Hendrycks, Zinan Lin, Yu Cheng, Sanmi Koyejo, Dawn Song, and Bo Li. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 31232–31339. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/63cb9921eecf51bfad27a99b2c53dd6d-Paper-Datasets_and_Benchmarks.pdf.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. September 2022. URL https://openreview.net/forum?id=1PL1NIMMrw.

P. C. Wason. Reasoning about a Rule. *Quarterly Journal of Experimental Psychology*, 20(3): 273–281, August 1968. ISSN 0033-555X. doi: 10.1080/14640746808400161. URL https://doi.org/10.1080/14640746808400161.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How Does LLM Safety Training Fail? In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 80079–80110. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/fd6613131889a4b656206c50a8bd7790-Paper-Conference.pdf.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pp. 24824–24837, Red Hook, NY, USA, November 2022. Curran Associates Inc. ISBN 978-1-7138-7108-8.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate Before Use: Improving Few-shot Performance of Language Models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12697–12706. PMLR, July 2021. URL https://proceedings.mlr.press/v139/zhao21c.html.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and Transferable Adversarial Attacks on Aligned Language Models, December 2023. URL http://arxiv.org/abs/2307.15043. arXiv:2307.15043 [cs].

A TASK VARIANCE DECOMPOSITION

For each task we sample $K > 1$ task variations and evaluate a model's binary success $N > 1$ times on each task variation. This leaves us with two sources of variance: the within-task-variation sampling noise and the between-task-variation variance due to the isomorphic task variations. Both of these sources can ultimately make the models appear unsystematic. We quantify their contribution to the total variance in the following.

A.1 THEORETICAL QUANTITIES

For a single fixed base task T, we define the following random variables:

- Let τ be a random variable representing the task variation $\tau(T)$.
- Let $\hat{p}_\tau$ be the empirical success rate of the model on task variation τ, calculated over N independent attempts.
- Let $p_\tau := \mathbb{E}[\hat{p}_\tau|\tau]$ be the true underlying success probability of the model on task variation τ.

To understand the total variance in the observed success rates, $\hat{p}_\tau$, across all possible task variations, we apply the law of total variance:

$$\mathbb{V}\mathrm{ar}(\hat{p}_\tau) = \mathbb{E}_\tau[\mathbb{V}\mathrm{ar}(\hat{p}_\tau|\tau)] + \mathbb{V}\mathrm{ar}_\tau(\mathbb{E}[\hat{p}_\tau|\tau]) \tag{1}$$

Because $\hat{p}_\tau$ is the sample mean of N independent Bernoulli trials, each with success probability p_τ, the variance of the empirical success rate given the task variation is $\mathbb{V}\mathrm{ar}(\hat{p}_\tau|\tau) = \frac{p_\tau(1-p_\tau)}{N}$. Substituting Equation (1) and rearranging the terms yields:

$$\mathbb{V}\mathrm{ar}_\tau(p_\tau) = \mathbb{V}\mathrm{ar}(\hat{p}_\tau) - \mathbb{E}_\tau\left[\frac{p_\tau(1-p_\tau)}{N}\right].$$

This equation allows us to isolate the true between-task-variation variance of the underlying success probabilities, $\mathbb{V}\mathrm{ar}_\tau(p_\tau)$, from the expected within-task-variation variance purely due to the finite sampling noise of N attempts per task variation, $\mathbb{E}_\tau\left[\frac{p_\tau(1-p_\tau)}{N}\right]$ using the total variation of the empirical success rate, $\mathbb{V}\mathrm{ar}(\hat{p}_\tau)$.

A.2 EMPIRICAL ESTIMATES

For a given task, we sample K task variations. Let $\hat{p}_{\tau_k}$ be the model's empirical success rate for sampled variation τ_k (where $k \in \{1, \ldots, K\}$), and $\hat{p}$ be the model's overall empirical success rate for this task across all $K \cdot N$ attempts.

We estimate the between-task-variation variance, $\hat{\sigma}^2_{\text{between}}$, of the underlying success probabilities across variations by removing the estimated within-task-variation variance, $\hat{\sigma}^2_{\text{within}}$ from the estimated total variance, $\hat{\sigma}^2_{\text{total}}$, as outlined in the derivation above:

$$\hat{\sigma}^2_{\text{between}} = \hat{\sigma}^2_{\text{total}} - \hat{\sigma}^2_{\text{within}}$$

We compute the following empirical estimators for the variance components:

Total observed variance We estimate the sample variance of the variation success rates as:

$$\hat{\sigma}^2_{\text{total}} = \frac{1}{K-1}\sum_{k=1}^{K}(\hat{p}_{\tau_k} - \hat{p})^2$$

Expected within-task-variation variance To prevent true variation differences from inflating the estimated noise, we use the following unbiased local estimator to calculate the expected binomial variance over N attempts,

$$\hat{\sigma}^2_{\text{within}} = \frac{1}{K}\sum_{k=1}^{K}\frac{\hat{p}_{\tau_k}(1-\hat{p}_{\tau_k})}{N-1}.$$

Normalization Since the empirical success rates differ between tasks, so do the theoretically obtainable maximum variances, which makes a comparison difficult. For this reason, we normalize the three variance components, expressing them as a proportion of the maximum possible variance. With each $\hat{p}_{\tau_k} \in [0, 1]$, the total observed sample variance is bounded by a theoretical maximum of $\sigma^2_{max} = \frac{K}{K-1}\hat{p}(1-\hat{p})$. We then obtain the

- Normalized total variance, $\rho_{\text{total}} = \frac{\hat{\sigma}^2_{\text{total}}}{\sigma^2_{\text{max}}}$,
- Normalized within-task-variation variance, $\rho_{\text{within}} = \frac{\hat{\sigma}^2_{\text{within}}}{\sigma^2_{\text{max}}}$,
- Normalized between-task-variation variance, $\rho_{\text{between}} = \frac{\hat{\sigma}^2_{\text{between}}}{\sigma^2_{\text{max}}}$.

A.3 Results

In Table 1, we report the average normalized variance components across task families, type of task variation and model. We find that the within-task-variation variance often makes up the largest part of the total variance.

Model	Metric	Task family							
		Integer list		Grammar-based instruction			Raven	Boolean concept	
		Substitution	Reordering	Recomposition	Reordering	Substitution	Rebinding	Substitution	Reordering
Claude Opus 4.7	ρ_{total}	0.20	0.17	0.12	0.17	0.20	0.62	-	-
	ρ_{within}	0.16	0.17	0.18	0.17	0.16	0.08	-	-
	ρ_{between}	0.04	0.00	0.00	0.01	0.04	0.54	-	-
GPT-5.5	ρ_{total}	0.19	0.15	0.17	0.17	-	0.19	-	-
	ρ_{within}	0.16	0.17	0.17	0.17	-	0.16	-	-
	ρ_{between}	0.03	0.00	0.00	0.00	-	0.03	-	-
Gemini 3.1 Flash	ρ_{total}	0.15	0.18	0.10	0.10	0.30	0.70	0.17	0.24
	ρ_{within}	0.17	0.16	0.19	0.19	0.15	0.06	0.17	0.15
	ρ_{between}	0.00	0.01	0.00	0.00	0.15	0.64	0.00	0.09
Gemini 3.1 Flash-Lite	ρ_{total}	0.22	0.36	0.11	0.23	0.05	0.37	0.19	0.15
	ρ_{within}	0.16	0.13	0.19	0.16	0.20	0.13	0.16	0.17
	ρ_{between}	0.07	0.23	0.00	0.07	0.00	0.25	0.03	0.00
Gemini 3.1 Pro	ρ_{total}	0.20	0.15	0.14	0.31	0.20	0.44	-	-
	ρ_{within}	0.16	0.17	0.18	0.14	0.17	0.11	-	-
	ρ_{between}	0.04	0.00	0.00	0.17	0.03	0.32	-	-

Table 1: **Variance decomposition across tasks and models.** Average estimated variance components for each model and task family grouped by the specific task variation type. Since our estimators are noisy, ρ_{within} can be larger than ρ_{total}, in which case we clip ρ_{between} to zero.

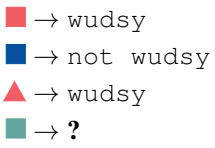
B Additional task family

B.1 Boolean concept learning

We adapt the Boolean concept learning task family studied in Feldman (2000) and Piantadosi et al. (2016). In each task, a learner must infer a latent concept from examples to classify objects with varying physical features, e.g. shapes with different colors and sizes into binary categories.

Instructions Figure 14 shows the general task instructions we provide in the system prompt asking models to identify the *simplest* rule to classify the support examples. What constitutes the *simplest* rule is underspecified without knowledge of the description language. For this reason, for each task we describe the context-free grammar from which the rule can be generated and with respect to which it is the shortest rule that explains the support set, see Figure 14 for an example.

Example The following is a sample episode from the Boolean concept learning task family.



To predict the query example of this task, the learner must infer that the simplest Boolean rule that explains the support set is that *all red objects are wudsy*.

Task structure We procedurally generate Boolean logic expressions of varying complexity from a probabilistic context-free grammar. Our grammar supports generating compositions of the logical operators `and` and `or`, as well as base predicates that evaluate specific features of the objects (e.g., `is_color(x, red)`. Notably, the grammar does not contain a `not` operator to reduce redundancy in the hypothesis space over possible rules. The particular features objects have in a specific task are a random subset of three out of five possible features (shape, size, color, material, and numerosity), where each feature can take three distinct values (e.g., red, blue, green).

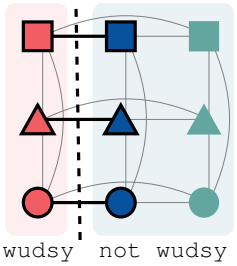


Figure 6: Example Hamming graph used to construct the support set from objects at the decision boundary (dashed line) of a given rule.

Support and query set When naively sampling a support set for a given rule, it is likely that a simpler rule than the one used to label the support examples can equally explain the support set. To contend with this possibility, we strategically build the support set and use an ideal observer model to ensure the rule used to generate the set is unique and the simplest to explain it. Specifically, to create the support set, we construct the Hamming graph over all theoretically possible objects, with an edge between any two objects that differ by exactly one feature as shown in Figure 6. As a heuristic, we can then select all pairs of adjacent objects in the Hamming graph that lie on the decision boundary of the given rule and randomly sample the query set from the remaining objects.

Task invariances The specific physical feature to which a rule applies should not matter to a systematic learner, i.e. the rule *all red objects are wudsy* is structurally equivalent to *all squares are wudsy*. We can therefore create isomorphic task variations both by permuting the order of the support and query samples (*example reordering*) as well as by rebinding the physical features of a rule and correspondingly changing the support and query set (*symbol substitution*).

C Additional results

Figure 8 shows additional results on the grammar-based instruction learning task family, Figure 9 shows additional results on the program induction over integer sequences task family, and Figure 10 shows additional results on the Symbolic Raven's progressive matrices task family, all described in Section 5 of the main text.

C.1 Reasoning effort does not consistently improve systematicity

Inference-time scaling with increased reasoning budgets typically improves model performance. While our main comparison in Figure 1 was conducted with reasoning effort set to high, we would like to understand the impact of the reasoning effort on systematicity. Apriori, one might expect that increasing the reasoning effort could reduce the impact of task variations on model variance and therefore possibly improve systematicity.

Figure 7 shows the systematicity metrics across task families for Gemini models when varying the reasoning effort from `low` to `medium`, and `high`. We restrict this analysis to Gemini models to keep API costs manageable. Surprisingly, increasing reasoning effort does not consistently improve systematicity. On the *program induction over integer sequences* and *symbolic Raven's progressive matrices* task families, we can observe a modest improvement in task performance but the systematicity gap largely remains unaffected. On the *grammar-based instruction-learning* task family the effect is highly model-dependent.

D Task instructions

For each task family, we provide detailed instructions in the system prompt. Figure 11 lists the instructions for the grammar-based instruction-learning task family, Figure 12 lists the instructions for the symbolic Raven's progressive matrices task family, and Figure 13 lists the instructions for the program induction over integer sequences task family.

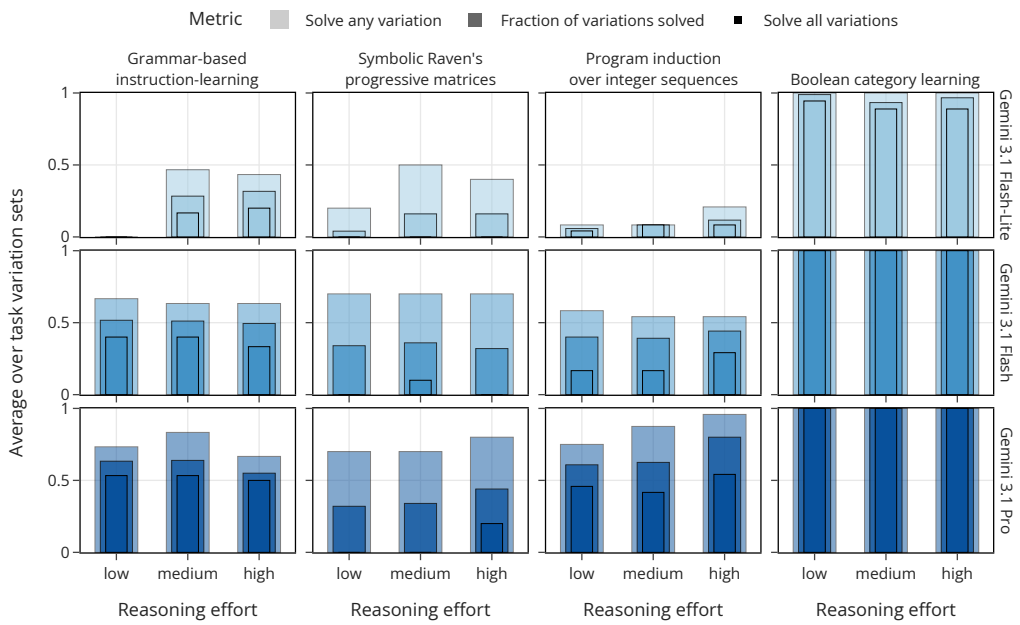


Figure 7: **Reasoning effort does not consistently improve systematicity.** Varying the reasoning effort for different Gemini reasoning models across task families does not consistently improve systematicity.

E Additional details

E.1 Software and libraries

For the results obtained in this paper we build on free and open-source software. We implemented our experiments in Python using NumPy (Harris et al., 2020) and JAX (Bradbury et al., 2018, Apache License 2.0). We run API models using Mozilla's any-llm library (Apache License 2.0), utilized WandB (Biewald, 2020, MIT license) to monitor the progress and results of experiments, and Plotly (Inc, 2015, MIT license) for generating the plots. We use uv for Python project dependency management (Marsh, 2024, MIT License).

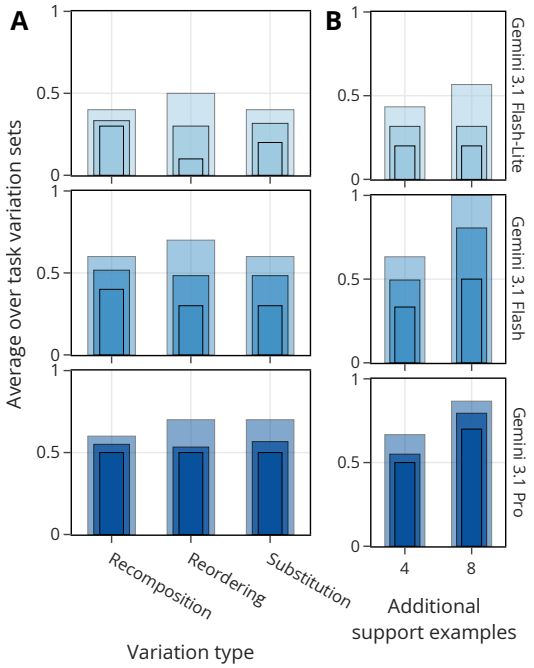


Figure 8: **Grammar-based instruction-learning.** (A) The systematicity gap persists across variation type. (B) Showing 8 rather than 4 support examples in addition to the curriculum examples increases the average number of tasks where any variation is solved but does not resolve the systematicity gap.

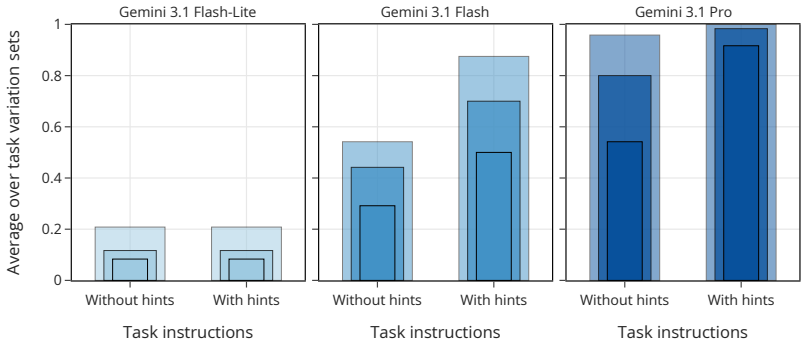


Figure 9: **Program induction over integer sequences.** Providing hints in the task instructions improves performance for Gemini 3.1 Flash and Gemini 3.1 Pro and almost eliminates the systematicity gap in the latter.

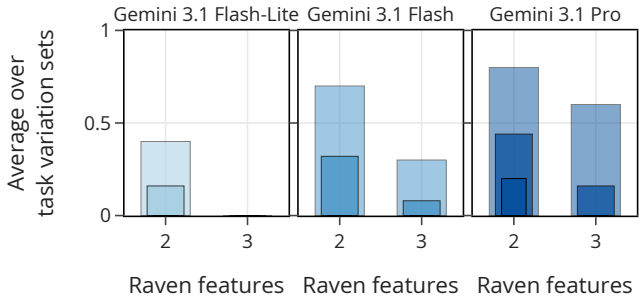


Figure 10: **Symbolic Raven's progressive matrices.** Increasing the number of features notably increases difficulty and leads to a corresponding drop in the average number of tasks where any variation can be solved.

```
Your task is to discover the grammatical rules of an artificial
language.  In the following, you will be presented with examples of
input and output string pairs.  These pairs demonstrate how sequences
of primitive tokens are transformed by grammatical rules.  Afterwards,
you will be presented with n_query new input sequence(s), and you
have to translate them into output sequences based on the grammatical
rules you have inferred.  Try to identify the simplest deterministic
rules that govern these transformations and apply them to the new
input sequences.
Every artificial language consists of:

1. Primitive rules:  Simple one-to-one mappings from an input token
   to an output token.

2. Function rules:  These operate on one or two arguments.  Functions
   are represented by specific tokens and apply to their adjacent
   arguments.

   • A 1-argument function appears after its argument (e.g., "arg1
     func").  A 2-argument function appears between its arguments
     (e.g., "arg1 func arg2").
   • Each argument is either a single primitive token or a sequence of
     tokens.
   • A function transforms its evaluated arguments into a new sequence
     by duplicating and/or rearranging them.
   • When the same 2-argument function is chained with itself, it
     evaluates left-to-right (e.g., "arg1 func arg2 func arg3" is
     evaluated as "[arg1 func arg2] func arg3") otherwise functions
     that accept sequences follow a strict priority order:  If Rule
     A evaluates before Rule B, and Rule B before Rule C, then Rule A
     always evaluates before Rule C.
```

Figure 11: Instructions for the grammar-based instruction-learning task family.

Your task is to predict integers in structured sequences of integers. Specifically, you will be presented with tasks that consist of three rows with three columns each. Each panel in this three by three matrix contains n_features feature(s) encoded as integer(s). Within a row the n_features feature(s) of the last column can be predicted by applying the correct rules to the first two columns. Within a task, all rows follow the same underlying rules applied to different inputs.
Each row is delimited by ||, each column is delimited by | and the features within a panel are separated by spaces. The last n_features feature(s) in the last panel are masked with n_features question marks, '?'.

1. Try to identify which integers in a given task are governed by the same underlying rules.
2. Try to identify the n_features rule(s) that govern a task.
3. Consider the following rules/patterns:
   - Constant, e.g. the same integer is repeated
   - Progression, e.g. integers are systematically incremented or decremented by steps of +1, +2, -1, or -2
   - Modular addition, e.g. two integers prior in the sequence are summed to produce the next integer
   - Modular subtraction, e.g. two integers prior in the sequence are subtracted to produce the next integer
   - Maximum/Minimum, e.g. the maximum/minimum of two integers prior in the sequence makes up the next integer

Figure 12: Instructions for the symbolic Raven's progressive matrices task family.

Your task is to discover the rule that transforms input lists of integers into output lists of integers. In the following, you will be presented with n_support examples of input and output list pairs. Afterwards, you will be presented with n_query new input list(s) and you have to apply the inferred rule to predict their corresponding output lists.

With hints

The rule is a composition of the following basic operations:

- insert: Insert an element at a specific position.
- remove: Remove an element at a specific position.
- repeat: Repeat the list until it reaches a specified length.
- shift: Shift all elements by a given offset, wrapping around.
- swap: Swap the first and second halves of the list.
- tail: Keep only a specified number of elements from the end of the list.

Figure 13: Instructions for the program induction over integer sequences task family.

```
Your task is to discover the meaning of 'wudsy', a word in an alien
language.  In the following you will be presented with several
examples of objects that are either wudsy or not wudsy.  Afterwards
you will be presented with n_query new object(s) and you have to
determine whether they are either wudsy or not wudsy.  Try to
identify the simplest, deterministic rule for what makes objects
wudsy or not wudsy and apply it to decide for the new objects.
Each rule is generated from a context-free grammar.  The specific
grammar and properties of the objects will be provided for each task.
The shortest generated rule that explains the examples is the correct
one to base the decision on.
Note in particular that this grammar does not contain a not()
operation, keep this in mind when trying to identify the "simplest"
rule, i.e.  the shortest rule that explains all examples that can be
expressed in this grammar.
Respond with the boolean value for each of the n_query new object(s)
in their presented order according to the rule you identified.
```

Example task-specific instruction

Each object has a shape, color and size.
The rules are generated from the following context-free grammar:

$S \to$ 'true' | 'false' | and(S, S) | or(S, S) | is_shape(O, A) | is_color(O, B) | is_size(O, C)

$O \to$ 'x'

$A \to$ 'circle' | 'triangle' | 'square'

$B \to$ 'blue' | 'yellow' | 'green'

$C \to$ 'tiny' | 'small' | 'large'

Figure 14: Instructions for the Boolean concept learning task family.