INDUCTIONBENCH: LLMS FAIL IN THE SIMPLEST COMPLEXITY CLASS

Wenyue Hua[1] **Tyler Wong**[1] **Fei Sun** **Liangming Pan**[2] **Adam Jardine**[3] **William Yang Wang**[1*]

[1]University of California, Santa Barbara, [2]University of Arizona,
[3]Rutgers University, New Brunswick

ABSTRACT

Large language models (LLMs) have shown remarkable improvements in reasoning and many existing benchmarks have been addressed by models such as o1 and o3 either fully or partially. However, a majority of these benchmarks emphasize deductive reasoning, including mathematical and coding tasks in which rules such as mathematical axioms or programming syntax are clearly defined, based on which LLMs can plan and apply these rules to arrive at a solution. In contrast, *inductive reasoning*, where one infers the underlying rules from observed data, remains less explored. Such inductive processes lie at the heart of scientific discovery, as they enable researchers to extract general principles from empirical observations. To assess whether LLMs possess this capacity, we introduce **InductionBench**, a new benchmark designed to evaluate the inductive reasoning ability of LLMs. Our experimental findings reveal that even the most advanced models available struggle to master the simplest complexity classes within the subregular hierarchy of functions, highlighting a notable deficiency in current LLMs' inductive reasoning capabilities. Coda and data are available `https://github.com/Wenyueh/inductive_reasoning_benchmark`.

1 INTRODUCTION

The remarkable progress of large language models (LLMs) in recent years has yielded substantial improvements in their reasoning capabilities. This progress is most evident in benchmarks involving complex mathematics (Cobbe et al., 2021; Hendrycks et al., 2021) and coding tasks (Jain et al., 2024; Jimenez et al., 2023; Chen et al., 2021; Fan et al., 2023). Beyond these domains, researchers have also explored the logical reasoning abilities of LLMs from various angles, including propositional logic (Zhu et al., 2023), first-order logic (Han et al., 2022; Parmar et al., 2024), and propositional logic under different contexts (Hua et al., 2024).

Despite significant progress in model capabilities, existing benchmarks predominantly focus on deductive reasoning, largely overlooking inductive reasoning. The former requires applying explicitly defined premises to derive valid conclusions, whereas the latter requires inferring the underlying principles, rules, or patterns from observations (Hawthorne, 2004). Both forms of reasoning are essential; inductive reasoning, in particular, is critical in domains such as scientific discovery where researchers seek to characterize natural laws based on empirical data (Grünwald, 2007; Hansen & Yu, 2001) that captures complex phenomena. Figure 1 illustrates the differences between inductive and deductive reasoning.

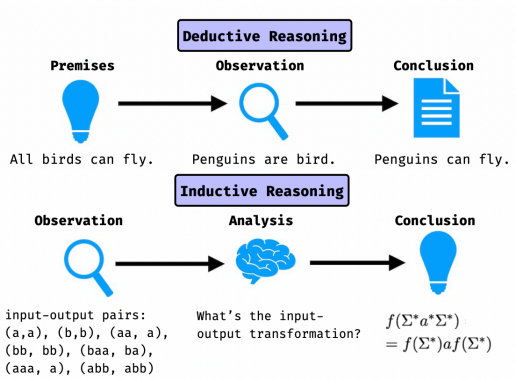

Figure 1: Deductive vs. Inductive Reasoning

In this paper, we address this gap by introducing **InductionBench**, a rigorous benchmark designed to assess LLMs' inductive reasoning abilities by testing whether they can infer a string-to-string transformation from a finite set of input–output pairs. A model must hypothesize the underlying relationship between inputs and outputs based on a finite set of examples and then extrapolate those rules to unseen strings. The process of discovering the underlying function from limited data reflects the core principles of inductive reasoning.

Our benchmark is grounded in the subregular hierarchy (Rogers & Pullum, 2011; Truthe, 2018; Graf, 2022; Jäger & Rogers, 2012; Heinz, 2018) (see Figure 2) for string-to-string mappings (Mohri, 1997), focusing on input transformations restricted to regular functions. By systematically increasing task complexity across multiple classes in the subregular hierarchy, we gain insights into how effectively LLMs detect, hypothesize, and generalize underlying rules from theoretically sufficient datapoints.

We evaluate multiple state-of-the-art LLMs to understand LLM's inductive reasoning ability and identify factors that increase the difficulty of inductive reasoning tasks for LLMs, such as the length of the minimum-length description, the number of datapoints, and in-context examples. Through extensive experiments, we find that even advanced models such as o3-mini struggle with basic inductive tasks, highlighting a significant shortcoming in the current generation of LLMs. More detailed findings are presented in Section 5.

2 Related Work

Deductive Reasoning. One major branch of reasoning benchmarks centers on deductive inference, where models apply established premises to derive specific conclusions. Notable examples include ReClor (Yu et al., 2020), which evaluates the ability to solve logical reasoning questions resembling those found in standardized tests, and various logic-based benchmarks of increasing complexity, from propositional logic to first-order logic (Han et al., 2022; Parmar et al., 2024; Zhu et al., 2023; Hua et al., 2024). These tasks typically require handling structured logical relationships with minimal ambiguity in how premises lead to conclusions.

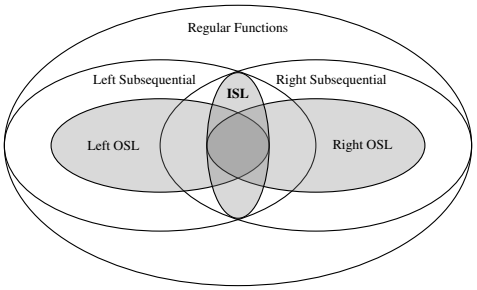

Figure 2: Subregular hierarchy in string-to-string maps

Another type of reasoning benchmarks is mathematical problem solving, including elementary arithmetic to advanced competition-level questions. Hendrycks et al. (2021) test both computational skills and the sequential reasoning steps involved in mathematics. Cobbe et al. (2021) covers a broad spectrum of topics, including geometry and higher-level problem solving. However, most standard mathematics problem-solving tasks can be framed as deductive reasoning, as they involve applying established axioms, definitions, and theorems in a logically valid sequence to derive a conclusion.

Inductive Reasoning. Despite the diversity of existing benchmarks, inductive reasoning remains comparatively underexplored where where models hypothesize and generalize patterns from examples without pre-specified rules. Current evaluations of inductive skills have largely been limited to small-scale symbolic regression, artificial language translation, and concept learning (Liu et al., 2024b; Lake et al., 2019; Qiu et al., 2023), which often lack three key elements: (1) an explicit analysis of the inherent difficulty of the task (2) a guarantee that the provided input–output dataset can identify the target function (3) a mechanism to evaluate whether models can identify the "best possible hypothesis" under Occam's Razor (Blumer et al., 1987; Baker, 2007) principle, *i.e.*, a description with minimal length (Hansen & Yu, 2001; Grünwald, 2007).

Our Contribution. To address these shortcomings, we introduce a new benchmark targeting on inductive reasoning skills. Building on subregular hierarchy and corresponding polynomial time and data learnability guarantees, our benchmark, **InductionBench**, tests how effectively LLMs infer underlying transformation functions from finite datapoints. We also measure the degree to

which models produce minimal, non-redundant hypotheses, providing a lens into their ability of generalization. Through a fine-grained, gradually increasing level of complexity, our evaluations reveal how current LLMs cope with the growing search space. Three main advantages are:

1. **Automated Evaluation**: Because the data is derived from well-defined functions, one can directly compare the model's output with the known ground-truth function, eliminating the need for expensive human annotations.
2. **Dynamic Data Generation**: The dataset is produced randomly based on specific function classes, allowing periodic "refreshes" to prevent models from relying on memorized examples.
3. **Rigorous Assessment of Hypothesis Space**: As the function is well-defined, one can control the size of the hypothesis space with precision. This control enables a rigorous and systematic evaluation of LLM performance from a theoretically grounded perspective.

3 Computational Complexity in Inductive Reasoning

InductionBench uses string-to-string transformation/functions as a proxy to study inductive reasoning, which has established computational complexity hierarchy(Roche & Schabes, 1997; Engelfriet & Hoogeboom, 2001). We focus on the subregular hierarchy, the hierarchy under regular functions. Though with limited expressive power, our experiments show that these classes already present substantial challenges for LLMs.

Specifically, we limit our attention to three classes of deterministic regular functions—*Left Output-Strictly-Local* (L-OSL), *Right Output-Strictly-Local* (R-OSL), and *Input-Strictly-Local* (ISL), whose positions in the subregular hierarchy are illustrated in Figure 2 (Heinz, 2018). These classes represent the lowest-complexity tier for string-to-string mappings within the subregular hierarchy. They are proper subclasses of subsequential function class and, more broadly, of weakly-deterministic class and non-deterministic class, which are themselves subsets of the regular function classes. Although we do not elaborate on the complete regular function hierarchy here, it is important to note that the ISL, L-OSL, and R-OSL classes are among the simplest in this framework.

Strictly local functions can be seen as operating with a fixed amount of look-ahead, similar to Markov processes. They are *provably learnable in polynomial time from polynomially sized samples* (Chandlee et al., 2014; De La Higuera, 1997; Chandlee et al., 2015; Jardine et al., 2014). Moreover, prior work has shown that an algorithm exists to learn the unique (up to isomorphism) smallest subsequential finite-state transducer that represents such ISL, L-OSL, R-OSL functions (Satta & Henderson, 1997; Arasu et al., 2009). This property allows us to evaluate not only whether LLMs can discover the correct patterns but also whether they can identify the simplest or most concise representation consistent with the data.

3.1 Preliminary

Before providing the definitions of the three function classes, we first introduce the fundamental mathematical notations and formal definitions underpinning our discussion of string-to-string transformations and their properties.

Let Σ be a finite alphabet. We denote by Σ^* the set of all finite strings over Σ, and by $\Sigma^{\leq n}$ the set of all strings over Σ of length at most n. The empty string is denoted as λ. The set of prefixes of a string w is denoted as $\textsc{Pref}(w)$, defined as $\{p \in \Sigma^* \mid \exists s \in \Sigma^* s.t. w = ps\}$, and the set of suffixes of w denoted as $\textsc{Suff}(w)$, defined as $\{s \in \Sigma^* \mid \exists p \in \Sigma^* s.t. w = ps\}$. The longest common prefix of a set of strings S is denoted as $\textsc{lcp}(S)$, defined as

$$p \in \cap_{w \in S} \textsc{Pref}(w) \text{ such as } \forall p' \in \cap_{w \in S} \textsc{Pref}(w), |p'| < |p|. \tag{1}$$

For any function $f : \Sigma^* \to \Gamma^*$ and $w \in \Sigma^*$, let the tails of w with respect to f be defined as

$$\textsc{tails}_f(w) = \{(y, v) \mid f(wy) = uv \text{ and } u = \textsc{lcp}(f(w\Sigma^*))\}. \tag{2}$$

Intuitively, $\textsc{tails}_f(w)$ collects all possible continuations (y, v) by appending y to w. It summarizes how f might extend beyond the partial input w. The total number of distinct tails across all strings in Σ^* provides a measure of how many different non-trivial local transformation f encodes.

3.2 Function Class Definition

Based on the concepts outlined above, we define the three function classes.

Definition 1 (ISL) *A function f is ISL if there is a k such that for all $u_1, u_2 \in \Sigma^*$, if* $\textsc{Suff}^{k-1}(u_1) = \textsc{Suff}^{k-1}(u_2)$*, then* $\textsc{tails}_f(u_1) = \textsc{tails}_f(u_2)$.

In simpler terms, this means that the output at each position in the string depends only on the preceding $k-1$ characters of the *input*, making the transformation *Markovian* with respect to the input. Below is a simple example:

Example 3.1 *Suppose a function $f : \{a, b\}^* \to \{a, b\}^*$ rewrites each b to a* only *if it appears after the input substring ba. In this scenario, we have $k = 3$, and there are two distinct* tails*:*

$$\textsc{tails}_f(w) = \{(\lambda, \lambda), (b, a), (bb, ab), (ab, ab) \dots\}, \quad \forall w \in \Sigma^* \textit{ such that } ba \in \textsc{suff}(w)$$

$$\textit{and } \textsc{tails}_f(w') = \{(\lambda, \lambda), (a, a), (bb, bb), (ab, ab) \dots\}, \quad \forall w' \in \Sigma^* \textit{ such that } ba \notin \textsc{suff}(w')$$

These tails indicate how the function's behavior shifts depending on whether the immediate context ends in ba. Such context-dependent tails also highlights that ISL functions can be effectively characterized or represented by local input constraints.

Definition 2 (L-OSL) *A function f is L-OSL if there is a k such that for all $u_1, u_2 \in \Sigma^*$, if* $\textsc{Suff}^{k-1}(f(u_1)) = \textsc{Suff}^{k-1}(f(u_2))$*, then* $\textsc{tails}_f(u_1) = \textsc{tails}_f(u_2)$.

In other words, the output at each position in the transformed string depends only on the preceding $k-1$ characters of the *output* itself, rather than on the input. This property can be understood as a form of Markovian process *on the output*. Below is a simple example:

Example 3.2 *Suppose a function f rewrites each b to λ* only *if it appears after the output substring ba. In this scenario, we have $k = 3$, and there are two distinct* tails*:*

$$\textsc{tails}_f(w) = \{(\lambda, \lambda), (a, a), (b, \lambda), (bb, \lambda), (ab, ab), (ba, a), \dots\} \quad \forall w \in \Sigma^* \textit{ such that } ba \in \textsc{suff}(f(w))$$

$$\textit{and } \textsc{tails}_f(w) = \{(\lambda, \lambda), (a, a), (b, b), (bb, bb), (ab, ab), (ba, ba) \dots\} \quad \forall w \in \Sigma^* \textit{ such that } ba \notin \textsc{suff}(f(w))$$

While L-OSL depends preceding output symbols to the "left", R-OSL functions depends on a limited number of *future* output symbols to the "right". Conceptually, one can view R-OSL as analogous to L-OSL, except that the input is processed in reverse order. Although both belong to the broader OSL paradigm, they are *incomparable* classes: each can express transformations the other cannot. The formal definition of R-OSL follows:

Definition 3 (R-OSL) *A function f is R-OSL if there is a k such that for all $u_1, u_2 \in \Sigma^*$, if* $\textsc{Suff}^{k-1}(f(u_1^{-1})) = \textsc{Suff}^{k-1}(f(u_2^{-1}))$*, then* $\textsc{tails}_f(u_1^{-1}) = \textsc{tails}_f(u_2^{-1})$.

Intuitively, this class of functions can be viewed as a *rightward* Markovian process on the output. Each output symbol is determined not by the preceding symbols as in L-OSL but by the next $k-1$ symbols that will appear in the output.

The three classes, ISL, L-OSL, and R-OSL, are each deterministic and exhibit Markovian behavior, yet remain pairwise incomparable within the broader subregular hierarchy. In this work, we further restrict our attention to functions that involve *substitution* which replaces one character with another and *deletion* which maps a character to the empty string λ.

3.3 Learnability

The three function classes are *identifiable in polynomial time using a polynomially sized characteristic sample* (Chandlee et al., 2014; 2015). In other words, there exists a polynomial-time algorithm that, given sufficient data for a target function f, can produce a representation τ that satisfies $f(w) = \tau(w)$ for every $w \in \Sigma^*$. In other words, once sufficient data is presented, one can reliably recover a function equivalent to f on all possible inputs. This learnability property underpins the value of these classes as testbeds for inductive reasoning, since the data requirement remains polynomial and successful inference is theoretically guaranteed.

We formalize "sufficient data" as the minimal set of input–output pairs needed to learn a k-strictly local function f, which is known as characteristic sample. Adapting the original definition[1] for clarity (Chandlee et al., 2014; 2015), we define:

Definition 4 (Characteristic Sample) *For a given k-ISL f, the characteristic sample S is defined as $\{(w, w') \mid w \in \Sigma^{\leq k} \wedge f(w) = w'\}$. For a given k-OSL f, the characteristic sample S is defined as $\{(w, w') \mid w' \in \Sigma^{\leq k} \wedge f(w) = w'\}$.*

If a provided dataset contains such characteristic sample, a learning algorithm can reconstruct a representation of f that matches its behavior on every string in Σ. Accordingly, in the context of LLMs, we expect that providing this dataset as in-context examples should enable the model to induce the underlying string-to-string mapping.

3.4 Unique Function Representation

Beyond verifying that a model can accurately discover a function from data, we also investigate how succinctly the model describes its inferred rules. This aspect is of both theoretical and practical interest: a minimal or *most concise* representation not only offers interpretability advantages but can also reflect the model's capacity for truly generalizable, rather than merely enumerative, learning.

One function can be represented or written in a non-unique way. For instance, consider an ISL function f_1 with $k = 2$ over $\Sigma = \{a, b\}$ that maps the input character a to b when it comes after b, that rewrites each a to b only if the preceding character is b, while leaving other substrings unchanged. One concise description is:

$$f_1(w) = \begin{cases} f_1(w_1)ba^{-1}f_1(aw_2), \text{if } w_1 \text{ ends with } b \text{ and } w{=}w_1\ aw_2 \text{ for some } w_1, w_2{\in}\Sigma^* \\ w, \quad \text{otherwise} \end{cases} \tag{3}$$

An alternative yet more verbose description of the same function might redundantly enumerate multiple cases:

$$f_1'(w) = \begin{cases} f_1'(w_1)ba^{-1}f_1'(aw_2), \text{if } w_1 \text{ ends with } ab \text{ and } w = w_1aw_2 \text{ for some } w_1, w_2 \in \Sigma^* \\ f_1'(w_1)ba^{-1}f_1'(aw_2) \text{if } w_1 \text{ ends with } bb \text{ and } w = w_1aw_2 \text{ for some } w_1, w_2 \in \Sigma^* \\ w, \quad \text{otherwise} \end{cases} \tag{4}$$

Although these two representations encode the same function, the second contains repetitive conditions and fails to emphasize that the output of f_1 depends solely on the single preceding character instead of the penultimate character.

Because these functions admit a *unique* minimal representation (up to isomorphism) (Chandlee et al., 2014; Oncina & Garcia, 1991), we can directly compare the function produced by an LLM to the ground-truth minimal form. In doing so, we evaluate whether the model not only *discovers* the correct transformation but also *simplifies* it to the most parsimonious description possible—an essential indicator of robust inductive reasoning.

3.5 Rule-based Representation

To streamline the generation and parsing of function representations, we employ a simplified notation wherein each transformation is written as "condition $\circ$ target character $\rightarrow$ output of the target character" (Bird & Ellison, 1994). In this notation, the *condition* represents the minimal substring needed to trigger a transformation, while any input substring not matching this condition remains unchanged. For instance, in the earlier example, this approach permits a concise notation $b \circ a \rightarrow b$, indicating that the input a is mapped to b when it comes after b; otherwise, the input string remains unaltered. This concise, rule-based format simplifies both the model's output generation (by reducing complex functional descriptions) and our subsequent evaluation, as the applicable transformations can be easily parsible and verified.

To demonstrate the simplicity of rule-based representation: given an ISL function f_2 with $k = 2$, the input a becomes b when it comes after b and two consecutive as will be reduced to one single a. The

[1] simplified from original definition

minimal function representation is as below:

$$f_2(w)=\begin{cases} f_2(w_1)ba^{-1}f_2(aw_2), \text{if } w_1 \text{ ends with } b \text{ and } w{=}w_1aw_2 \text{ for some } w_1, w_2{\in}\Sigma^* \\ f_2(w_1)a^{-1}f_2(aw_2), \text{if } w_1 \text{ ends with } a \text{ and } w{=}w_1aw_2 \text{ for some } w_1, w_2{\in}\Sigma^* \\ w, \quad \text{otherwise} \end{cases} \tag{5}$$

In the simplified rule-based format, it can be written as:$a \circ a \to \lambda$, $b \circ a \to b$. In summary, f_1 can be minimally expressed with a single rule, f_2 requires two rules.

4 Benchmark Construction

In this section, we detail how our benchmark is constructed from the previously defined function classes. Each datapoint $(\mathcal{D}, f)$ in the benchmark is a pair of dataset $\mathcal{D}$ and function f where $\mathcal{D}$ is a set of input-output pairs generated by f.

Each of ISL, L-OSL, and R-OSL classes can be further subdivided into incremental levels of complexity, determined by three key parameters: (1) the context window size k (2) the vocabulary size $|\Sigma|$ (3) the minimal representation length of the function, *i.e.* the minimal set of rules corresponding to the function. Given k and $|\Sigma|$, the search space is $2^{|\Sigma|^k}$; given the number of rules n additionally, the search space is $\binom{|\Sigma|^k}{n}$. To rigorously evaluate LLMs' inductive capabilities, we systematically vary these parameters across ISL, L-OSL, and R-OSL function classes.

In addition, we examine how performance changes with different numbers of input–output pairs in the prompt. Although having the characteristic sample present should theoretically guarantee recoverability of the underlying function, our empirical results indicate that the overall number of examples strongly affects performance. While extra data can provide richer information, it also increases context length considerably and heightens processing demands (Li et al., 2024). By varying the number of provided datapoints, we further investigate the extent to which the model engages in genuine reasoning and how robust its inductive abilities remain under changing input sizes.

Function Generation To systematically create benchmark instances, we first *randomly generate* functions f based on the three parameters: k, $|\Sigma|$, and the number of minimal rules describing f by generating the set of rules that can describe f. While multiple representations of varying length can describe the same function, each function has a *unique minimal representation* (up to isomorphism). During function generation, we therefore ensure that each function is expressed by a minimal, non-redundant rule set. Formally, if a f is represented by a set of rules $R_f = \{r_1, r_2, ..., r_n\}$ where each r_i has the form of $c_i \circ u_i \to v_i$ (with c_i as the condition substring, u_i the target character, and v_i the transformed output for u_i), there are several constraints may be applied to functions belonging to the three classes.

Definition 5 (General Consistency) *Given f represented by a set of rules $R_f : \forall r_i, r_j \in R_f, c_i \circ u_i \notin \textsc{Suff}(c_j \circ u_j)$ and $c_j \circ u_j \notin \textsc{Suff}(c_i \circ u_i)$.*

General Consistency ensures that the rules do not contradict one another or become redundant when conditions overlap. For instance, a function whose rule-based representation of $r_1 : a \circ b \to a$ and $r_2 : aa \circ b \to a$ is redundant, as the scenarios where r_1 is applied is a superset of the scenarios where r_2 is applied. For another instance, there does not exist a deterministic function that can be described by $r_1 : a \circ b \to a$ and $r_2 : aa \circ b \to \lambda$. Generating rule-based representations for ISL functions needs only satisfy this constraint.

Definition 6 (OSL Non-Redundancy Guarantee) *Given f represented by a set of rules $R : \forall r_i \in R_f, \neg\exists s'_i \in\{s_i|s_i{\in}c_i\}$ such that $\exists r_j \in R_f$ such that $s'_i = c_j \circ u_j$, unless $\exists r_k \in R$ such that $c_k \circ v_k = s'_i$.*

Constraint 2 is specific to the two OSL function classes because we need to make sure that all output conditions in the rule actually surface somewhere in the outputs of some datapoints. If the output condition c never actually surface as the output, the rule will never be put into effect. Thereby the above rule basically requires that condition part of all rules can surface, either because it will never be modified by some other rule, or it emerges on the surface because of the application of other rule. For

instance, a function represented by rules $r_1 : aa \circ b \to a, r_2 : a \circ a \to c$ is redundant because r_1 will never be applied because the string aa will never surface as output and thus it will never be put into effect; For another instance, a function represented by $r_1 : aa \circ b \to a, r_2 : a \circ a \to c, r_3 : a \circ d \to a$ is non redundant because even though into aa string will be modified into ac, but aa will surface in some datapoint because ad will be modified into aa and thus r_1 will be able to be applied.

Generating the functions following the two constraints, we ensure that the generated function representation is minimal, non-reducible guarantees a clear measure of complexity. One additional requirement is imposed to ensure each function indeed requires a look-ahead of size k. Specifically:

Definition 7 (**k-Complexity Guarantee**) *Given f whose designated context window $k = k_1$, $\exists r' \in R$ such that $c' \circ u' \to v'$ such that $|c' \circ u'| = k_1$.*

This condition guarantees that the function is genuinely k-strictly local (for ISL or OSL), rather than being representable with a smaller window size. Consequently, the functions we generate faithfully reflect the intended complexity level.

After generating the function f, we generate the characteristic sample of input-output pairs. For instance, given a function f with $k = 2$ and $\Sigma = \{a, b\}$, the characteristic sample is $\{(a, f(a)), (b, f(b)), (ab, f(ab)), (aa, f(aa)), (bb, f(bb)), (ba, f(ba))\}$, a small set whose size is 6. By expanding this sample set, we can explore whether providing more than the minimal necessary examples aids or hinders the model's performance to infer the underlying function.

To evaluate how effectively an LLM can induce the underlying function, we include in the prompt (1) the function class, (2) context window k, (3) the alphabet Σ which are information that guarantee learnability of the function. Then given the sample dataset, we request LLMs to produce a minimal rule-based description that reproduces the provided sample set, revealing whether it can *discover* and *optimally represent* the underlying transformation.

5 Main Experiment

Experiment Setting We evaluate using zero-shot chain-of-thought prompting on six SOTA LLMs, including Llama-3.3-70b (Dubey et al., 2024) with FP8 quantization, Llama-3.1-405b with FP8 quantization, GPT-4o (Hurst et al., 2024), DeepSeek-V3 (Liu et al., 2024a), o1-mini(Jaech et al., 2024), and o3-mini. For all models, we evaluate with all settings including $k \in \{2, 3, 4\}$, $|\Sigma| \in \{2, 3, 4\}$, number of rules $\in \{2, 3, 4\}$, and sample set size to be 1, 2, 3, 4 times larger than the characteristic sample. For each setting, we randomly generate 10 functions f and corresponding input-output sample $\mathcal{D}$ to calculate the result. As o1-mini and o3-mini perform much better than other models, in addition, we evaluate on two more complex settings with $k \in \{4, 5\}, |\Sigma| = 5$.

Evaluation Metrics For each experiment setting, we leverage three metrics to evaluate performance: Precision, Recall, Compatibility. Let R be the unique ground-truth rule set of minimal length for function f, P be the predicted rule set generated by LLM, $\mathcal{D}$ be the provided sample set in the context on which we evaluate the correctness of P.

Precision measures how many of the predicted rules are correct relative to all rules the model generated: $\frac{|R \cap P|}{|P|}$. captures the proportion of the model's rules that align exactly with the ground-truth rules. A higher precision indicates fewer unnecessary/redundant rules.

Recall measures what fraction of the ground-truth rules the model successfully recovered: $\frac{|R \cap P|}{|R|}$. A higher recall reflects the model's ability to cover all aspects of the correct transformation.

Compatibility measures whether applying the predicted rule set P to each input in the sample set $\mathcal{D}$ yields the correct output:

$$\text{Compatibility}(P, \mathcal{D}) = \begin{cases} 1 & \text{if } \forall (x_i, y_i) \in \mathcal{D}, P(x_i) = y_i \\ 0 & \text{otherwise} \end{cases}$$

Compatibility is the most fundamental measure, as it verifies whether the generated function accurately reproduces all observed input–output pairs in $\mathcal{D}$. A trivial way to achieve perfect compatibility is to include every pair $(x_i, y_i[-1]) \in \mathcal{D}$ as an independent rule; however, doing so results in very low precision, indicating a failure to capture the underlying generalizable structure of the function. Note that all results presented are expressed as percentages.

5.1 MODEL PERFORMANCE COMPARISON

Models	ISL			L-OSL			R-OSL		
	recall	precision	compatibility	recall	precision	compatibility	recall	precision	compatibility
Llama-3.3 70B	10.00	5.32	0.00	10.00	6.33	0.00	10.00	10.83	0.00
Llama-3.1 405B	10.00	3.75	0.00	6.67	1.10	0.00	13.33	1.85	0.00
GPT-4o	10.00	2.67	0.00	13.33	3.82	0.00	16.67	6.73	0.00
DeepSeek-V3	13.33	2.46	0.00	23.33	2.73	0.00	3.33	0.25	0.00
o1-mini	36.67	22.09	0.00	43.33	32.12	0.00	26.67	17.58	0.00
o3-mini	**73.33**	**59.58**	**10.00**	**66.67**	**69.17**	**10.00**	**63.33**	**62.00**	**30.00**

Table 1: Zero-shot CoT benchmark result with $k = 4$, $|\Sigma|$ = 4, number of rules = 3, sample size = 2

Table 1 showcases model performance under particularly challenging conditions: $k = 4$, $|\Sigma| = 4$ with 3 rules and sample size twice that of the minimal size of characteristic sample (detailed analysis on the impact of sample size is presented in Section 5.2). As seen in the table, compatibility scores collapse to 0 for all models except o3-mini, which also achieves relatively modest compatibility overall. This pattern highlights the difficulty that current LLMs face when required to track slightly broader contexts window even under very limited vocabulary size = 4. Full results are presented in Tables 3, 4, 5 in Appendix. Table 2 further reports the performance of o1-mini and o3-mini under slightly more challenging settings. Although both models generally exhibit non-trivial recall and precision, their compatibility scores consistently remain at or near zero. It is important to note that ISL, L-OSL, and R-OSL are the simplest function classes within the subregular hierarchy of string-to-string mappings. Thus, despite the strong performance of state-of-the-art models on benchmarks in coding (Jain et al., 2024), mathematics (Mirzadeh et al., 2024), and knowledge-intensive tasks (Wang et al., 2024), they falter on this elementary inductive reasoning task.

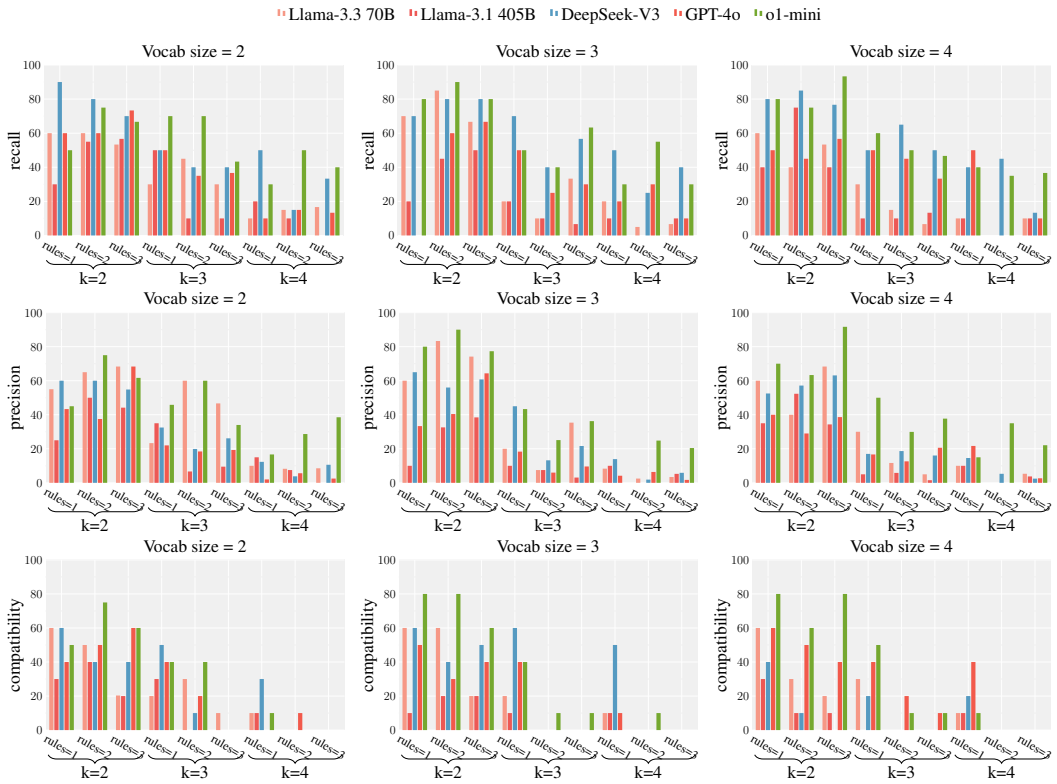

Figure 3: ISL results for five models: Llama3.3-70b, Llama3.1-405b, GPT-4o, DeepSeek-V3, o1-mini

5.2 IMPACT OF DIFFERENT FACTORS

Models	Settings	k = 4			k = 5		
		R	P	C	R	P	C
ISL							
o1-mini	rules = 4	25.00	12.21	0.00	10.00	9.10	0.00
	rules = 5	36.00	40.41	0.00	10.00	3.14	0.00
o3-mini	rules = 4	37.50	49.83	0.00	27.50	30.75	0.00
	rules = 5	42.00	58.67	0.00	20.00	38.33	0.00
L-OSL							
o1-mini	rules = 4	32.50	37.34	0.00	15.00	29.33	10.00
	rules = 5	28.00	23.17	0.00	8.00	4.61	0.00
o3-mini	rules = 4	57.50	58.93	0.00	22.50	39.26	0.00
	rules = 5	48.00	71.38	0.00	10.00	23.67	0.00
R-OSL							
o1-mini	rules = 4	17.50	22.33	0.00	12.50	7.63	0.00
	rules = 5	16.00	15.42	0.00	18.00	22.82	0.00
o3-mini	rules = 4	45.00	43.76	10.00	20.00	50.00	0.00
	rules = 5	38.00	55.17	0.00	14.00	36.17	0.00

Table 2: o1-mini and o3-mini results (R = Recall, P = Precision, C = Compatibility) on harder setting with $k \in \{4, 5\}$, $|\Sigma| = 5$, sample size = 2

Figure 3 shows how five models (Llama3.3-70B, Llama3.1-405B, GPT-4o, DeepSeek-V3, and o1-mini) perform on various ISL tasks, organized by three key parameters: the context window size k, the vocabulary size $|\Sigma|$, and the minimal number of rules required to describe the function. The top row of panels presents recall, the middle row presents precision, and the bottom row presents compatibility. We did not include o3-mini here because its performance is way stronger than all other five models and thereby in order to show the impact of various factors, we omit this model for better visual clarity. Based on these figures, the impact of k, $|\Sigma|$, and number of rules become very clear: (1) increasing k markedly reduces recall, precision, and compatibility (2) $|\Sigma|$ does not impact the performance much (3) the number of minimal rules reduces compatibility with large k and $|\Sigma|$.

Impact of k. Across all models, moving from $k = 2$ to $k = 4$ markedly reduces recall, precision, and compatibility. This trend underscores how increasing the context window increases the complexity of the underlying ISL functions and making it more challenging for current LLMs to learn the correct transformations. Longer look-ahead requires the model to track additional input context, which can overload its capacity to induce reliable rules.

Impact of $|\Sigma|$. In contrast, enlarging the vocabulary from $|\Sigma| = 2$ to $|\Sigma| = 4$ does not consistently degrade performance to the same degree as increasing k. While some models exhibit slight declines in recall or precision with a larger alphabet, these effects are neither as uniform nor as pronounced as those induced by a bigger Markov window. This finding suggests that the breadth of symbol variation matters less than the depth of sequential dependencies.

Impact of the Number of Rules. Notably, the number of minimal rules can substantially affect compatibility. When $k = 2$ and $|\Sigma| = 2$, a comparatively small search space, changing the number of rules does not drastically alter compatibility. However, under more demanding scenarios where $k \in \{3, 4\}$, the data indicate that adding rules can cause compatibility to plummet. In many cases, having just one rule still yields nontrivial compatibility, whereas introducing a second or third rule often overwhelms the models, resulting in compatibility scores near 0.

Impact of Examples We further examined whether few-shot prompting could enhance model performance. In our experiments with Llama-3.3 70B, we varied the number of in-context examples (1-shot, 2-shot, and 3-shot) to determine their effect on the model's ability to induce the correct function representation. The results indicate that when both the vocabulary size and the context window k are small, adding more examples improves performance across the evaluated metrics. However, as the complexity increases—with larger values of k and vocabulary sizes—the benefits of additional few-shot examples become negligible. These findings suggest that while few-shot learning is beneficial for simpler settings, its efficacy diminishes in more complex inductive tasks. Experiment results are presented in Tables 6, 7, and 8.

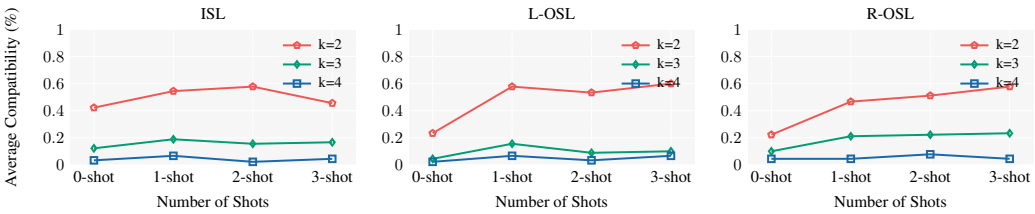

Figure 4: Impact of Few-shot Examples under different k.

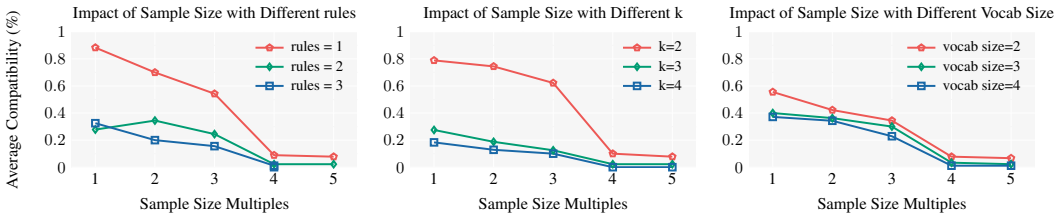

Figure 5: Impact of Sample Size with Different Sample Size Multiples.

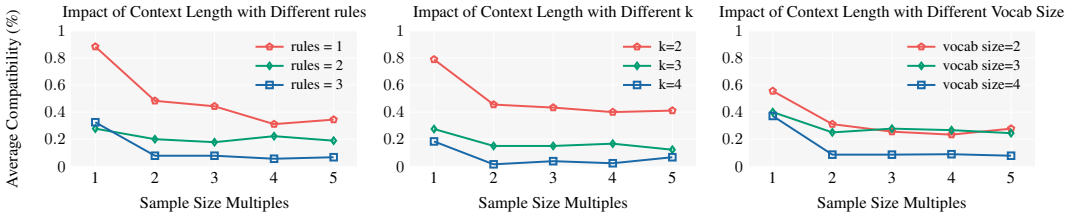

Figure 6: Impact of Context Length with Different Sample Size Multiples.

Robustness We assess the stability of inductive reasoning by varying the number of input–output pairs provided to the model. The x-axis represent $\frac{|\mathcal{D}|}{|S|}$ where S is the minimal set of examples needed to guarantee learnability of the underlying function. The hypothesis is that if the model were performing genuine logical or inductive reasoning, we would expect performance to remain stable or even improve as more data points become available, since these points should further clarify the underlying function. Figure 5 illustrates how average compatibility decreases steeply as the number of provided input–output examples increases. This drop suggests that the LLM's reasoning process is not robustly inductive: rather than refining its hypothesis with additional data, the model appears to become confused or overwhelmed, leading to poorer overall performance. Consequently, these findings highlight the limited robustness of current LLMs' inductive reasoning, particularly in scenarios where increasing the available data should theoretically facilitate, rather than hinder, function inference.

Moreover, to isolate the influence of context length from the effect of adding genuinely new data, we conduct an additional experiment in which we simply extending the context size by *repeating* the minimal characteristic sample without introducing novel input–output pairs. Comparing Figures 5 and 6 reveals that while compatibility does diminish with increased context length (*e.g.*, at a multiple of 2), the decline is relatively small when scaling further to multiples of 3, 4, or 5. By contrast, when truly new datapoints are added (and not just repeated), compatibility plummets nearly to zero for multiples of 4 and 5. These results confirm that the primary driver of performance degradation is the inclusion of additional, distinct datapoints rather than simply lengthening the context. Further Analysis on error types are presented in Appendix B.

Leaderboard built based on the function classes are presented in Appendix C.

6 Conclusion

In this work, we introduced a systematic benchmark for assessing the inductive reasoning capabilities of LLMs, leveraging both well-studied subregular function classes and a more exploratory class (Details in Appendix C) for which no known polynomial-time learning algorithm exists. By controlling parameters such as the Markov window size k, the vocabulary size $|\Sigma|$, and the minimal number of rules, we offered precise yet flexible tasks capable of probing a model's capacity to infer general transformations from limited data. Our findings revealed several significant challenges for current LLMs, especially when required to track deeper dependencies or manage larger search spaces, and underscored the fragility of their inductive reasoning under increased context or novel data. Overall, our results highlight the need for more robust inductive reasoning strategies within current LLM architectures. We hope that our benchmark will help catalyze progress in both theoretical understanding and practical innovations around LLMs' inductive capabilities.

REFERENCES

Arvind Arasu, Surajit Chaudhuri, and Raghav Kaushik. Learning string transformations from examples. *Proceedings of the VLDB Endowment*, 2(1):514–525, 2009.

Alan Baker. Occam's razor in science: a case study from biogeography. *Biology & Philosophy*, 22 (2):193–215, 2007.

Steven Bird and T Mark Ellison. One-level phonology: Autosegmental representations and rules as finite automata. *Computational Linguistics*, 20(1):55–90, 1994.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information processing letters*, 24(6):377–380, 1987.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics*, 2:491–504, 2014.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. Output strictly local functions. In *14th Meeting on the Mathematics of Language*, pp. 112–125, 2015.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Colin De La Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138, 1997.

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

Joost Engelfriet and Hendrik Jan Hoogeboom. Mso definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic (TOCL)*, 2(2):216–254, 2001.

Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. Nphardeval: Dynamic benchmark on reasoning ability of large language models via complexity classes. *arXiv preprint arXiv:2312.14890*, 2023.

Thomas Graf. Diving deeper into subregular syntax. *Theoretical Linguistics*, 48(3-4):245–278, 2022.

Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, et al. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.

Mark H Hansen and Bin Yu. Model selection and the principle of minimum description length. *Journal of the american statistical association*, 96(454):746–774, 2001.

James Hawthorne. Inductive logic. 2004.

Jeffrey Heinz. The computational nature of phonological generalizations. *Phonological typology, phonetics and phonology*, 23:126–195, 2018.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Wenyue Hua, Kaijie Zhu, Lingyao Li, Lizhou Fan, Shuhang Lin, Mingyu Jin, Haochen Xue, Zelong Li, JinDong Wang, and Yongfeng Zhang. Disentangling logic: The role of context in large language model reasoning capabilities. *arXiv preprint arXiv:2406.02787*, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Gerhard Jäger and James Rogers. Formal language theory: refining the chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1956–1970, 2012.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Adam Jardine, Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. Very efficient learning of structured classes of subsequential functions from positive data. In *International Conference on Grammatical Inference*, pp. 94–108. PMLR, 2014.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Brenden M Lake, Tal Linzen, and Marco Baroni. Human few-shot learning of compositional instructions. *arXiv preprint arXiv:1901.04587*, 2019.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Emmy Liu, Graham Neubig, and Jacob Andreas. An incomplete loop: Deductive, inductive, and abductive learning in large language models. *arXiv preprint arXiv:2404.03028*, 2024b.

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

Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311, 1997.

José Oncina and Pedro Garcia. Inductive learning of subsequential functions. *Univ. Politécnica de Valencia, Tech. Rep. DSIC II*, 31, 1991.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13679–13707, 2024.

Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, et al. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. *arXiv preprint arXiv:2310.08559*, 2023.

Emmanuel Roche and Yves Schabes. *Finite-state language processing*, volume 115. MIT press Cambridge, MA, 1997.

James Rogers and Geoffrey K Pullum. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20:329–342, 2011.

Giorgio Satta and John Henderson. String transformation learning. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 444–451, 1997.

Bianca Truthe. Hierarchy of subregular language families. 2018.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. Reclor: A reading comprehension dataset requiring logical reasoning. *arXiv preprint arXiv:2002.04326*, 2020.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. Dyval: Dynamic evaluation of large language models for reasoning tasks. In *The Twelfth International Conference on Learning Representations*, 2023.

A Appendix

Full results on ISL, OSL, and few-shot experiments are presented here.

Models	Settings	k = 2			k = 3			k = 4		
		recall	precision	compatibility	recall	precision	compatibility	recall	precision	compatibility
vocab size = 2										
Llama-3.3 70B	rules = 1	60.00	55.00	60.00	30.00	23.33	20.00	10.00	10.00	10.00
	rules = 2	60.00	65.00	50.00	45.00	60.00	30.00	15.00	8.25	0.00
	rules = 3	53.33	68.33	20.00	30.00	46.67	10.00	16.67	8.54	0.00
Llama-3.1 405B	rules = 1	30.00	25.00	30.00	50.00	35.00	30.00	20.00	15.00	10.00
	rules = 2	55.00	50.00	40.00	10.00	6.67	0.00	10.00	7.50	0.00
	rules = 3	56.67	44.17	20.00	10.00	9.50	0.00	0.00	0.00	0.00
DeepSeek-V3	rules = 1	90.00	60.00	60.00	50.00	32.50	50.00	50.00	12.33	30.00
	rules = 2	80.00	60.00	40.00	40.00	19.89	10.00	15.00	3.82	0.00
	rules = 3	70.00	54.83	40.00	40.00	26.15	0.00	33.33	10.61	0.00
GPT-4o	rules = 1	60.00	43.33	40.00	50.00	22.00	40.00	10.00	2.00	0.00
	rules = 2	60.00	37.50	50.00	35.00	18.43	20.00	15.00	5.63	10.00
	rules = 3	73.33	68.33	60.00	36.67	19.30	0.00	13.33	2.50	0.00
o1-mini	rules = 1	50.00	45.00	50.00	70.00	45.83	40.00	30.00	16.67	10.00
	rules = 2	75.00	75.00	75.00	70.00	60.00	40.00	50.00	28.67	0.00
	rules = 3	66.67	61.67	60.00	43.33	34.00	0.00	40.00	38.52	0.00
o3-mini	rules = 1	100.00	100.00	100.00	100.00	100.00	100.00	80.00	58.33	40.00
	rules = 2	90.00	90.00	90.00	90.00	90.67	90.00	85.00	80.00	50.00
	rules = 3	**100.00**	**97.50**	**100.00**	**83.33**	**71.83**	**60.00**	**70.00**	**63.81**	**20.00**
vocab size = 3										
Llama-3.3 70B	rules = 1	70.00	60.00	60.00	20.00	20.00	20.00	20.00	8.33	10.00
	rules = 2	85.00	83.33	60.00	10.00	7.50	0.00	5.00	2.50	0.00
	rules = 3	66.67	74.17	20.00	33.33	35.36	0.00	6.67	3.43	0.00
Llama-3.1 405B	rules = 1	20.00	10.00	10.00	20.00	10.00	10.00	10.00	10.00	10.00
	rules = 2	45.00	32.58	20.00	10.00	7.50	0.00	0.00	0.00	0.00
	rules = 3	50.00	38.45	20.00	6.67	3.10	0.00	10.00	5.27	0.00
DeepSeek-V3	rules = 1	70.00	65.00	60.00	70.00	45.00	60.00	50.00	13.93	50.00
	rules = 2	80.00	56.00	40.00	40.00	13.23	0.00	25.00	1.89	0.00
	rules = 3	80.00	60.76	50.00	56.67	21.64	0.00	40.00	5.88	0.00
GPT-4o	rules = 1	50.00	33.33	50.00	50.00	18.33	40.00	20.00	4.17	10.00
	rules = 2	60.00	40.42	30.00	25.00	6.00	0.00	30.00	6.39	0.00
	rules = 3	66.67	64.33	40.00	30.00	9.61	0.00	10.00	1.72	0.00
o1-mini	rules = 1	80.00	80.00	80.00	50.00	43.33	40.00	30.00	0.00	0.00
	rules = 2	90.00	90.00	80.00	40.0	25.11	10.00	55.00	24.82	10.00
	rules = 3	80.00	77.33	60.00	63.33	36.25	10.00	30.00	20.44	0.00
o3-mini	rules = 1	100.00	100.000	100.00	100.00	95.00	90.00	90.00	78.33	70.00
	rules = 2	100.00	100.000	100.00	95.00	91.67	80.00	75.00	75.00	50.00
	rules = 3	**96.67**	**97.50**	**80.00**	**93.33**	**91.67**	**90.00**	**83.33**	**85.17**	**50.00**
vocab size = 4										
Llama-3.3 70B	rules = 1	60.00	60.00	60.00	30.00	30.00	30.00	10.00	10.00	10.00
	rules = 2	40.00	40.00	30.00	15.00	11.67	0.00	0.00	0.00	0.00
	rules = 3	53.33	68.33	20.00	6.67	5.00	0.00	10.00	5.32	0.00
Llama-3.1 405B	rules = 1	40.00	35.00	30.00	10.00	5.00	0.00	10.00	10.00	10.00
	rules = 2	75.00	52.33	10.00	10.00	5.83	0.00	0.00	0.00	0.00
	rules = 3	40.00	34.33	10.00	13.33	1.54	0.00	10.00	3.75	0.00
DeepSeek-V3	rules = 1	80.00	52.50	60.00	50.00	17.00	40.00	40.00	14.58	40.00
	rules = 2	85.00	57.15	50.00	65.00	18.66	20.00	45.00	5.30	0.00
	rules = 3	76.67	63.12	40.00	50.00	16.05	10.00	13.33	2.46	0.00
GPT-4o	rules = 1	50.00	40.00	40.00	50.00	16.67	20.00	50.00	21.67	20.00
	rules = 2	45.00	29.00	10.00	45.00	12.62	0.00	0.00	0.00	0.00
	rules = 3	56.67	38.62	0.00	33.33	20.60	0.00	10.00	2.67	0.00
o1-mini	rules = 1	80.00	70.00	80.00	60.00	50.00	50.00	40.00	15.00	10.00
	rules = 2	75.00	63.33	60.00	50.00	29.93	10.00	35.00	35.00	0.00
	rules = 3	93.33	91.67	80.00	46.67	37.72	10.00	36.67	22.09	0.00
o3-mini	rules = 1	100.00	100.00	100.00	100.00	95.00	100.00	60.00	60.00	60.00
	rules = 2	100.00	100.00	100.00	95.00	91.67	80.00	75.00	76.67	40.00
	rules = 3	**96.67**	**95.00**	**90.00**	**93.33**	**93.33**	**80.00**	**73.33**	**59.58**	**10.00**

Table 3: Input Strictly Local with sample size = 2

B Error Analysis and Summary

Error Type Analysis We further examined the specific types of errors made by LLMs when their predicted functions failed to match the ground-truth dataset. At a high level, we distinguish between *missing rules* (leading to low recall) and *wrong rules* (leading to low precision).

Missing Rules: These refer to ground-truth rules that do not appear in the model's predicted rule set. We classify missing rules into three subtypes:

Models	Settings	k = 2			k = 3			k = 4		
		recall	precision	compatibility	recall	precision	compatibility	recall	precision	compatibility
vocab size = 2										
Llama-3.3 70B	rules = 1	50.00	45.00	50.00	0.00	0.00	0.00	0.00	0.00	0.00
	rules = 2	25.00	25.00	20.00	10.00	8.33	10.00	5.00	10.00	0.00
	rules = 3	56.67	65.00	0.00	6.67	8.33	0.00	13.33	12.83	0.00
Llama-3.1 405B	rules = 1	70.00	45.83	70.00	30.00	9.33	10.00	10.00	1.67	10.00
	rules = 2	50.00	33.33	10.00	25.00	11.39	0.00	10.00	3.00	0.00
	rules = 3	63.33	53.83	0.00	10.00	6.67	0.00	6.67	5.00	0.00
GPT-4o	rules = 1	30.00	12.50	30.00	30.00	10.83	10.00	10.00	5.00	10.00
	rules = 2	75.00	63.17	60.00	20.00	7.42	0.00	15.00	6.35	0.00
	rules = 3	66.67	60.00	50.00	30.00	19.00	10.00	10.00	4.16	0.00
DeepSeek-V3	rules = 1	100.00	75.00	70.00	50.00	32.50	40.00	40.00	12.78	40.00
	rules = 2	60.00	44.17	30.00	10.00	15.00	10.00	20.00	11.94	0.00
	rules = 3	83.33	77.67	50.00	20.00	12.92	0.00	23.33	13.97	0.00
o1-mini	rules = 1	90.00	90.00	90.00	70.00	55.00	40.00	10.00	10.00	10.00
	rules = 2	80.00	80.00	80.00	60.00	60.00	50.00	65.00	53.83	10.00
	rules = 3	90.00	82.50	50.00	66.67	60.67	20.00	50.00	54.22	10.00
o3-mini	rules = 1	100.00	100.00	100.00	90/00	90.00	90.00	90.00	90.00	90.00
	rules = 2	100.00	100.00	100.00	95.00	100.00	100.00	85.00	78.33	70.00
	rules = 3	**100.00**	**100.00**	**100.00**	**86.67**	**85.00**	**80.00**	**56.67**	**50.33**	**40.00**
vocab size = 3										
Llama-3.3 70B	rules = 1	50.00	50.00	50.00	20.00	12.50	10.00	20.00	13.33	10.00
	rules = 2	35.00	33.67	10.00	20.00	6.93	10.00	25.00	15.00	0.00
	rules = 3	40.00	65.00	20.00	20.00	18.33	0.00	10.00	2.78	0.00
Llama-3.1 405B	rules = 1	60.00	45.00	40.00	10.00	3.33	10.00	10.00	1.11	0.00
	rules = 2	30.00	20.00	0.00	15.00	13.33	0.00	5.00	0.53	0.00
	rules = 3	66.67	57.83	30.00	20.00	8.39	0.00	10.00	2.36	0.00
GPT-4o	rules = 1	40.00	27.50	40.00	20.00	8.33	20.00	40.00	11.00	30.00
	rules = 2	55.00	46.50	10.00	45.00	25.67	0.00	30.00	6.50	10.00
	rules = 3	60.00	50.00	10.00	33.33	15.95	0.00	20.00	6.21	0.00
DeepSeek-V3	rules = 1	80.00	70.00	60.00	50.00	22.00	40.00	50.00	16.11	30.00
	rules = 2	90.00	60.32	60.00	70.00	13.82	20.00	30.00	5.04	0.00
	rules = 3	66.67	53.50	40.00	23.33	16.42	0.00	30.00	7.54	**10.00**
o1-mini	rules = 1	100.00	95.00	90.00	80.00	63.33	70.00	30.00	17.50	30.00
	rules = 2	90.00	83.33	80.00	70.00	49.42	40.00	35.00	29.52	20.00
	rules = 3	**96.67**	**96.00**	**90.00**	70.00	56.15	30.00	50.00	33.58	0.00
o3-mini	rules = 1	100.00	100.00	100.00	90.00	90.00	90.00	80.00	80.00	80.00
	rules = 2	100.00	100.00	100.00	90.00	75.15	70.00	80.00	72.50	50.00
	rules = 3	**96.67**	94.17	**90.00**	**96.67**	**87.50**	**90.00**	**63.33**	**68.43**	**40.00**
vocab size = 4										
Llama-3.3 70B	rules = 1	50.00	29.00	30.00	20.00	13.33	10.00	10.00	10.00	10.00
	rules = 2	50.00	50.00	10.00	20.00	15.96	0.00	0.00	0.00	0.00
	rules = 3	50.00	52.50	20.00	6.67	6.00	0.00	10.00	6.33	0.00
Llama-3.1 405B	rules = 1	60.00	34.50	30.00	10.00	5.00	10.00	10.00	5.00	10.00
	rules = 2	50.00	29.00	0.00	10.00	3.13	0.00	10.00	2.90	0.00
	rules = 3	43.33	30.73	20.00	16.67	5.83	0.00	6.67	1.10	0.00
GPT-4o	rules = 1	40.00	35.00	30.00	60.00	25.33	40.00	40.00	12.50	10.00
	rules = 2	75.00	45.50	30.00	55.00	20.47	10.00	20.00	9.44	0.00
	rules = 3	70.00	48.22	20.00	33.33	10.77	0.00	13.33	3.82	0.00
DeepSeek-V3	rules = 1	100.00	82.50	80.00	50.00	23.67	30.00	40.00	16.11	40.00
	rules = 2	70.00	50.67	30.00	25.00	8.01	0.00	15.00	3.24	0.00
	rules = 3	60.00	48.36	30.00	50.00	12.81	20.00	23.33	2.73	0.00
o1-mini	rules = 1	90.00	85.00	90.00	50.00	38.33	30.00	40.00	28.33	20.00
	rules = 2	100.00	93.33	80.00	60.00	36.92	20.00	50.00	31.92	10.00
	rules = 3	80.00	72.25	60.00	70.00	43.04	10.00	43.33	32.12	0.00
o3-mini	rules = 1	100.00	100.00	100.00	100.00	100.00	100.00	60.00	60.00	60.00
	rules = 2	100.00	100.00	100.00	85.00	81.67	70.00	45.00	58.33	20.00
	rules = 3	**100.00**	**100.00**	**100.00**	**76.67**	**76.67**	**70.00**	**66.67**	**69.17**	**10.00**

Table 4: Left Output Strictly Local with sample size = 2

1. *Too General.* Although a certain ground-truth rule $r : c \circ u \rightarrow v$ was missed, there exists a corresponding predicted rule $r' : c' \circ u' \rightarrow v'$ that over-generalizes. Specifically, the condition c' is a *proper suffix* of c, causing r' to apply more broadly than intended.

2. *Too Specific.* The opposite of the above: a predicted rule condition c' is a proper *extension* of c, thus applying too narrowly and failing to match some instances that should have been captured by the ground-truth rule.

Models	Settings	k = 2			k = 3			k = 4		
		recall	precision	compatibility	recall	precision	compatibility	recall	precision	compatibility
vocab size = 2										
Llama-3.3 70B	rules = 1	40.00	40.00	40.00	20.00	10.00	10.00	20.00	10.00	10.00
	rules = 2	40.00	40.00	30.00	15.00	18.33	10.00	20.00	18.67	10.00
	rules = 3	63.33	76.67	30.00	23.33	24.17	0.00	10.00	7.83	0.00
Llama-3.1 405B	rules = 1	20.00	8.33	0.00	40.00	18.33	0.00	0.00	0.00	0.00
	rules = 2	60.00	51.67	40.00	25.00	12.50	0.00	10.00	6.25	10.00
	rules = 3	63.33	54.72	30.00	20.00	14.33	10.00	6.67	4.50	0.00
GPT-4o	rules = 1	50.00	30.00	30.00	10.00	5.00	10.00	0.00	0.00	0.00
	rules = 2	70.00	61.67	60.00	45.00	16.93	10.00	30.00	15.83	20.00
	rules = 3	83.33	71.83	30.00	43.33	24.10	0.00	20.00	10.00	0.00
DeepSeek-V3	rules = 1	60.00	38.33	20.00	40.00	14.17	20.00	20.00	7.00	10.00
	rules = 2	75.00	51.67	40.00	35.00	18.33	0.00	25.00	13.00	20.00
	rules = 3	86.67	72.56	50.00	40.00	27.00	0.00	23.33	3.68	0.00
o1-mini	rules = 1	50.00	38.33	40.00	20.00	5.83	0.00	20.00	11.67	10.00
	rules = 2	55.00	38.33	30.00	35.00	17.50	0.00	15.00	8.70	0.00
	rules = 3	46.67	46.67	10.00	33.33	24.17	0.00	10.00	4.41	0.00
o3-mini	rules = 1	90.00	90.00	90.00	100.00	95.00	90.00	70.00	50.00	30.00
	rules = 2	100.00	100.00	100.00	90.00	85.00	60.00	45.00	41.67	20.00
	rules = 3	**96.67**	**92.67**	**80.00**	**86.67**	**78.33**	**50.00**	**46.67**	**46.67**	**30.00**
Llama-3.3 70B	rules = 1	40.00	40.00	40.00	30.00	25.00	30.00	30.00	7.26	0.00
	rules = 2	30.00	60.00	10.00	20.00	10.93	0.00	25.00	19.17	10.00
	rules = 3	30.00	45.00	0.00	26.67	22.67	10.00	10.00	4.58	0.00
Llama-3.1 405B	rules = 1	20.00	11.43	10.00	10.00	5.00	0.00	20.00	4.17	0.00
	rules = 2	30.00	20.83	10.00	15.00	4.17	0.00	10.00	3.41	0.00
	rules = 3	16.67	9.11	0.00	10.00	4.75	0.00	0.00	0.00	0.00
GPT-4o	rules = 1	60.00	50.00	40.00	50.00	27.50	20.00	50.00	15.33	20.00
	rules = 2	60.00	41.67	30.00	50.00	27.50	20.00	30.00	8.82	0.00
	rules = 3	66.67	57.83	30.00	40.00	21.88	0.00	13.33	6.33	0.00
DeepSeek-V3	rules = 1	80.00	60.83	60.00	50.00	29.17	40.00	40.00	9.42	20.00
	rules = 2	80.00	63.19	50.00	55.00	26.67	20.00	40.00	10.10	0.00
	rules = 3	70.00	42.95	50.00	23.33	15.28	0.00	20.00	3.56	0.00
o1-mini	rules = 1	60.00	60.00	60.00	20.00	13.33	10.00	20.00	13.33	10.00
	rules = 2	15.00	15.00	10.00	40.00	25.00	0.00	30.00	19.50	0.00
	rules = 3	53.33	45.83	20.00	30.00	17.63	0.00	13.33	8.43	0.00
o3-mini	rules = 1	90.00	90.00	90.00	100.00	95.00	100.00	50.00	40.00	30.00
	rules = 2	90.00	83.33	70.00	80.00	61.67	30.00	60.00	59.50	40.00
	rules = 3	**100.00**	**100.00**	**100.00**	**83.33**	**64.11**	**50.00**	**43.33**	**41.83**	**20.00**
vocab size = 4										
Llama-3.3 70B	rules = 1	40.00	25.00	30.00	40.00	23.33	20.00	10.00	10.00	10.00
	rules = 2	45.00	47.33	10.00	50.00	45.83	10.00	5.00	1.67	0.00
	rules = 3	33.33	39.50	10.00	30.00	23.85	0.00	10.00	10.83	0.00
Llama-3.1 405B	rules = 1	10.00	10.00	10.00	0.00	0.00	0.00	0.00	0.00	0.00
	rules = 2	15.00	12.00	0.00	10.00	13.33	0.00	5.00	0.26	0.00
	rules = 3	33.33	22.67	0.00	3.33	16.67	0.00	13.33	1.85	0.00
GPT-4o	rules = 1	70.00	49.17	50.00	60.00	19.50	50.00	50.00	23.10	10.00
	rules = 2	80.00	78.10	40.00	40.00	16.02	0.00	20.00	10.00	0.00
	rules = 3	66.67	43.00	0.00	20.00	11.02	0.00	16.67	6.73	0.00
DeepSeek-V3	rules = 1	80.00	65.00	70.00	50.00	18.83	20.00	60.00	19.77	40.00
	rules = 2	70.00	59.17	30.00	45.00	27.79	20.00	10.00	0.88	0.00
	rules = 3	73.33	60.17	40.00	43.33	13.45	0.00	3.33	0.25	0.00
o1-mini	rules = 1	70.00	53.33	40.00	30.00	18.33	10.00	40.00	40.00	40.00
	rules = 2	70.00	66.67	50.00	50.00	47.50	20.00	40.00	34.00	0.00
	rules = 3	90.00	79.17	50.00	23.33	23.33	0.00	26.67	17.58	0.00
o3-mini	rules = 1	90.00	90.00	90.00	100.00	93.33	90.00	50.00	50.00	50.00
	rules = 2	100.00	100.00	100.00	80.00	75.00	60.00	70.00	69.17	40.00
	rules = 3	**100.00**	**100.00**	**100.00**	**76.67**	**78.33**	**50.00**	**63.33**	**62.00**	**30.00**

Table 5: Right Output Strictly Local with sample size = 2

3. *Completely Missed.* No predicted rule over-generalizes or under-generalizes the ground-truth rule; in other words, this pattern is simply absent from the predicted rule set altogether.

Wrong Rules: These refer to rules present in the model's predicted set that do not exist in the ground truth. We categorize such rules into four types:

Models	Settings	k = 2			k = 3			k = 4		
		recall	precision	compatibility	recall	precision	compatibility	recall	precision	compatibility
vocab size = 2										
rules = 1	0-shot	60.00	55.00	60.00	30.00	23.33	20.00	10.00	10.00	10.00
	1-shot	60.00	60.00	60.00	50.00	35.00	40.00	10.00	5.00	0.00
	2-shot	70.00	70.00	70.00	70.00	50.00	60.00	20.00	10.00	10.00
	3-shot	80.00	80.00	80.00	60.00	55.00	60.00	10.00	5.00	10.00
rules = 2	0-shot	60.00	65.00	50.00	45.00	60.00	30.00	15.00	8.25	0.00
	1-shot	65.00	70.00	60.00	40.00	53.00	30.00	20.00	18.33	10.00
	2-shot	85.00	85.00	80.00	45.00	56.67	20.00	25.00	23.33	0.00
	3-shot	60.00	65.00	40.00	35.00	36.67	10.00	20.00	20.00	0.00
rules = 3	0-shot	53.33	68.33	20.00	30.00	46.67	10.00	16.67	8.54	0.00
	1-shot	76.67	83.33	60.00	43.33	57.67	0.00	20.00	18.33	0.00
	2-shot	86.67	86.67	60.00	26.67	28.33	0.00	13.33	16.67	0.00
	3-shot	90.00	93.33	70.00	46.67	52.50	20.00	16.67	21.17	0.00
vocab size = 3										
rules = 1	0-shot	70.00	60.00	60.00	20.00	20.00	20.00	20.00	8.33	10.00
	1-shot	90.00	90.00	90.00	50.00	50.00	50.00	30.00	30.00	30.00
	2-shot	70.00	70.00	70.00	30.00	20.00	20.00	10.00	10.00	10.00
	3-shot	40.00	40.00	40.00	40.00	35.00	40.00	30.00	18.33	20.00
rules = 2	0-shot	85.00	83.33	60.00	10.00	7.50	0.00	5.00	2.50	0.00
	1-shot	90.00	95.00	80.00	10.00	13.33	0.00	5.00	2.50	0.00
	2-shot	65.00	65.00	40.00	30.00	28.33	10.00	5.00	2.00	0.00
	3-shot	65.00	75.00	30.00	5.00	3.33	0.00	5.00	2.50	0.00
rules = 3	0-shot	66.67	74.17	20.00	33.33	35.36	0.00	6.67	3.43	0.00
	1-shot	70.00	69.17	40.00	20.00	22.50	0.00	10.00	13.33	0.00
	2-shot	76.67	76.67	50.00	33.33	43.33	0.00	10.00	13.33	0.00
	3-shot	60.00	60.83	10.00	23.33	31.67	0.00	16.67	19.76	0.00
vocab size = 4										
rules = 1	0-shot	60.00	60.00	60.00	30.00	30.00	30.00	10.00	10.00	10.00
	1-shot	40.00	40.00	40.00	50.00	31.67	50.00	20.00	13.33	20.00
	2-shot	70.00	57.00	60.00	30.00	25.00	30.00	10.00	5.00	0.00
	3-shot	60.00	60.00	60.00	20.00	15.00	20.00	10.00	10.00	10.00
rules = 2	0-shot	40.00	40.00	30.00	15.00	11.67	0.00	0.00	0.00	0.00
	1-shot	60.00	60.00	30.00	15.00	23.33	0.00	0.00	0.00	0.00
	2-shot	80.00	90.00	60.00	15.00	12.50	0.00	15.00	10.67	0.00
	3-shot	70.00	70.00	70.00	15.00	18.33	0.00	10.00	10.00	0.00
rules = 3	0-shot	53.33	68.33	20.00	6.67	5.00	0.00	10.00	5.32	0.00
	1-shot	66.67	70.83	30.00	30.00	47.50	0.00	3.00	5.00	0.00
	2-shot	66.67	73.33	30.00	30.00	55.00	0.00	3.33	3.33	0.00
	3-shot	60.00	71.67	10.00	20.00	39.24	0.00	0.00	0.00	0.00

Table 6: Input Strictly Local with sample size = 2 with few-shot example

1. *Too General.* The rule $r' : c' \circ u' \rightarrow v'$ is overly broad, applying in contexts where the ground truth does not. This typically arises when c' is a proper suffix of some genuine condition c and thus fails to capture necessary constraints.
2. *Too Specific.* The rule narrowly addresses only a subset of the intended patterns (e.g., by employing a condition c' that is an extension of the legitimate condition c), thereby missing broader contexts that should have matched.
3. *Correct Condition but Wrong Transformation.* Here, the predicted rule accurately identifies the correct condition c' and target input character u', but the transformation v' is incorrect.
4. *Completely Wrong.* None of the above criteria apply: the rule's condition and transformation are both inconsistent with the ground truth, indicating a fundamental misunderstanding.

We present a breakdown of error types for three models in Figure7: Llama3.3-70B, o1-mini, and o3-mini. Among *missing rules*, the most common issue is **completely missed**, where the model fails to identify the relevant pattern at all. The second most frequent error is **too general**, suggesting that the predicted condition is shorter than needed, thereby overgeneralizing the intended behavior. In contrast, **too specific** errors in this category are relatively rare. Among *wrong rules*, the majority are **completely wrong**, followed by a notable fraction of **too general**. Although there is also a

Models	Settings	k = 2			k = 3			k = 4		
		recall	precision	compatibility	recall	precision	compatibility	recall	precision	compatibility
vocab size = 2										
rules = 1	0-shot	50.00	45.00	50.00	0.00	0.00	0.00	0.00	0.00	0.00
	1-shot	80.00	80.00	80.00	40.00	33.33	30.00	20.00	5.00	20.00
	2-shot	80.00	75.00	70.00	30.00	25.00	30.00	30.00	13.33	10.00
	3-shot	80.00	80.00	80.00	20.00	15.00	20.00	20.00	15.00	20.00
rules = 2	0-shot	25.00	25.00	25.00	10.00	8.33	10.00	5.00	10.00	0.00
	1-shot	80.00	85.00	70.00	30.00	30.83	10.00	30.00	19.00	10.00
	2-shot	85.00	85.00	80.00	20.00	21.67	10.00	25.00	27.90	0.00
	3-shot	75.00	80.00	60.00	25.00	20.83	10.00	20.00	20.83	0.00
rules = 3	0-shot	56.67	65.00	0.00	6.67	8.33	0.00	13.33	12.83	0.00
	1-shot	80.00	80.00	80.00	40.00	42.00	0.00	10.00	11.67	0.00
	2-shot	80.00	75.00	70.00	33.33	48.33	0.00	13.33	28.33	0.00
	3-shot	80.00	80.00	80.00	33.33	39.17	0.00	16.67	26.67	0.00
vocab size = 3										
rules = 1	0-shot	50.00	50.00	50.00	20.00	12.50	10.00	20.00	13.33	10.00
	1-shot	100.00	100.00	100.00	40.00	23.33	40.00	0.00	0.00	0.00
	2-shot	70.00	70.00	70.00	30.00	23.33	20.00	10.00	5.00	0.00
	3-shot	80.00	75.00	80.00	20.00	20.00	20.00	20.00	20.00	20.00
rules = 2	0-shot	35.00	33.67	10.00	20.00	6.93	10.00	25.00	15.00	0.00
	1-shot	80.00	76.67	80.00	30.00	30.33	10.00	10.00	15.00	0.00
	2-shot	50.00	55.00	30.00	30.00	38.33	0.00	25.00	21.67	0.00
	3-shot	70.00	70.00	70.00	20.00	30.00	0.00	20.00	35.00	0.00
rules = 3	0-shot	40.00	65.00	20.00	20.00	18.33	0.00	10.00	2.78	0.00
	1-shot	70.00	66.67	40.00	30.00	24.83	0.00	10.00	20.30	10.00
	2-shot	83.33	90.00	60.00	33.33	40.83	0.00	30.00	43.33	0.00
	3-shot	70.00	78.33	30.00	23.33	44.50	0.00	16.67	18.33	0.00
vocab size = 4										
rules = 1	0-shot	50.00	29.00	30.00	20.00	13.33	10.00	10.00	10.00	10.00
	1-shot	50.00	50.00	50.00	50.00	50.00	50.00	20.00	15.00	20.00
	2-shot	60.00	60.00	60.00	10.00	10.00	10.00	20.00	20.00	20.00
	3-shot	20.00	20.00	20.00	30.00	25.00	30.00	20.00	20.00	20.00
rules = 2	0-shot	50.00	50.00	10.00	20.00	15.96	0.00	0.00	0.00	0.00
	1-shot	55.00	56.67	30.00	20.00	28.33	0.00	0.00	0.00	0.00
	2-shot	55.00	50.00	20.00	35.00	55.00	10.00	5.00	10.00	0.00
	3-shot	10.00	20.00	0.00	30.00	33.33	10.00	10.00	20.00	0.00
rules = 3	0-shot	50.00	52.50	20.00	6.67	6.00	0.00	10.00	6.33	0.00
	1-shot	56.67	68.33	20.00	20.00	36.25	0.00	16.7	22.83	0.00
	2-shot	66.67	67.50	50.00	16.67	26.67	0.00	6.67	15.00	0.00
	3-shot	3.33	3.33	0.00	23.33	40.83	0.00	3.33	3.33	0.00

Table 7: Left Output Strictly Local with sample size = 2 with few-shot example

non-negligible number of **too specific** errors, these tend to occur when a single ground-truth rule (e.g., $ab \circ c \rightarrow b$ is replaced by multiple subcases (*e.g.*, $aab \circ c \rightarrow b, bab \circ c \rightarrow b, cab \circ c \rightarrow b$, indicating the model has uncovered individual instances but failed to unify them into a concise representation. Finally, **correct condition but wrong transformation** occurs relatively infrequently, implying that once the model infers the correct condition pattern, it typically produces the correct transformation.

B.1 Summary of findings

Overall, our experiments reveal four main insights into the inductive reasoning performance of current LLMs:

- Context window size k dominates complexity: Increasing k from 2 to 4 significantly degrades recall, precision, and compatibility, underscoring how longer look-ahead windows intensify the complexity of ISL functions.
- Number of Rules increases difficulty under large hypothesis space: The number of minimal rules required can drastically lower compatibility in more challenging settings with large k adn $|\Sigma|$, indicating that managing multiple interacting rules overwhelms many models.

Models	Settings	k = 2			k = 3			k = 4		
		recall	precision	compatibility	recall	precision	compatibility	recall	precision	compatibility
vocab size = 2										
rules = 1	0-shot	40.00	40.00	40.00	20.00	10.00	10.00	20.00	10.00	10.00
	1-shot	60.00	60.00	60.00	50.00	40.00	50.00	30.00	18.33	10.00
	2-shot	60.00	60.00	60.00	40.00	35.00	40.00	30.00	25.00	20.00
	3-shot	60.00	60.00	60.00	50.00	40.00	50.00	10.00	2.50	10.00
rules = 2	0-shot	40.00	40.00	30.00	15.00	18.33	10.00	20.00	18.67	10.00
	1-shot	60.00	60.00	50.00	40.00	42.50	20.00	30.00	33.33	0.00
	2-shot	70.00	80.00	60.00	45.00	43.33	30.00	20.00	27.50	0.00
	3-shot	90.00	90.00	90.00	45.00	41.67	30.00	15.00	15.00	0.00
rules = 3	0-shot	63.33	76.67	30.00	23.33	24.17	0.00	10.00	7.83	0.00
	1-shot	60.00	60.00	60.00	40.00	47.83	0.00	20.00	24.17	0.00
	2-shot	60.00	60.00	60.00	50.00	48.33	20.00	16.67	18.33	0.00
	3-shot	60.00	60.00	60.00	43.33	46.67	10.00	16.67	25.00	0.00
vocab size = 3										
rules = 1	0-shot	40.00	40.00	40.00	30.00	25.00	30.00	30.00	7.26	0.00
	1-shot	70.00	65.00	60.00	40.00	35.00	40.00	20.00	15.00	10.00
	2-shot	80.00	75.00	70.00	50.00	50.00	50.00	30.00	18.33	30.00
	3-shot	70.00	65.00	60.00	40.00	35.00	40.00	20.00	15.00	10.00
rules = 2	0-shot	30.00	60.00	10.00	20.00	10.93	0.00	25.00	19.17	10.00
	1-shot	65.00	70.00	40.00	30.00	45.00	10.00	15.00	11.67	0.00
	2-shot	60.00	70.00	40.00	50.00	44.17	10.00	20.00	20.00	0.00
	3-shot	65.00	70.00	40.00	30.00	45.00	10.00	15.00	11.67	0.00
rules = 3	0-shot	30.00	45.00	0.00	26.67	22.67	10.00	10.00	4.58	0.00
	1-shot	80.00	80.83	40.00	30.00	37.59	0.00	13.33	11.00	0.00
	2-shot	73.33	71.67	30.00	26.67	34.50	0.00	33.33	47.00	0.00
	3-shot	80.00	80.83	40.00	30.00	37.60	0.00	13.33	11.00	0.00
vocab size = 4										
rules = 1	0-shot	40.00	25.00	30.00	40.00	23.33	20.00	10.00	10.00	10.00
	1-shot	60.00	60.00	60.00	50.00	31.67	50.00	10.00	10.00	10.00
	2-shot	80.00	68.33	80.00	60.00	33.33	30.00	20.00	20.00	20.00
	3-shot	70.00	70.00	70.00	40.00	19.50	40.00	20.00	20.00	20.00
rules = 2	0-shot	45.00	47.33	10.00	50.00	45.83	10.00	5.00	1.67	0.00
	1-shot	80.00	70.00	40.00	45.00	61.67	10.00	5.00	10.00	0.00
	2-shot	65.00	65.00	40.00	45.00	52.50	20.00	0.00	0.00	0.00
	3-shot	75.00	75.00	70.00	45.00	48.33	20.00	15.00	30.00	0.00
rules = 3	0-shot	33.33	39.50	10.00	30.00	23.85	0.00	10.00	10.83	0.00
	1-shot	66.67	71.83	30.00	23.33	30.83	0.00	3.33	2.50	0.00
	2-shot	83.33	86.50	50.00	26.67	37.50	0.00	13.33	28.83	0.00
	3-shot	76.67	80.00	50.00	40.00	51.67	10.00	13.33	20.00	0.00

Table 8: Right Output Strictly Local with sample size = 2 with few-shot example

- Few-shot examples do not help much: Few-shot examples help in simpler configurations but yield diminishing returns as k adn $|\Sigma|$ grows—suggesting that, past a certain complexity threshold, additional examples do not compensate for the model's limited inductive capacity.

- Current LLMs are very unrobust: Providing more novel data should theoretically clarify function patterns, yet performance often plummets, reflecting a fragility in inductive reasoning.

- Error analysis shows that missing rules are most frequently "completely missed" or "too general," while wrong rules often end up "completely wrong" or again "too general." Only a small fraction are "too specific" or feature a "correct condition but wrong transformation," indicating that once models identify the right condition, they typically produce the correct transformation.

Taken together, these findings highlight fundamental limits in current LLMs' inductive reasoning. Even state-of-the-art models often fail as complexity grows, or when confronted with more data than their inductive mechanisms appear able to systematically absorb.

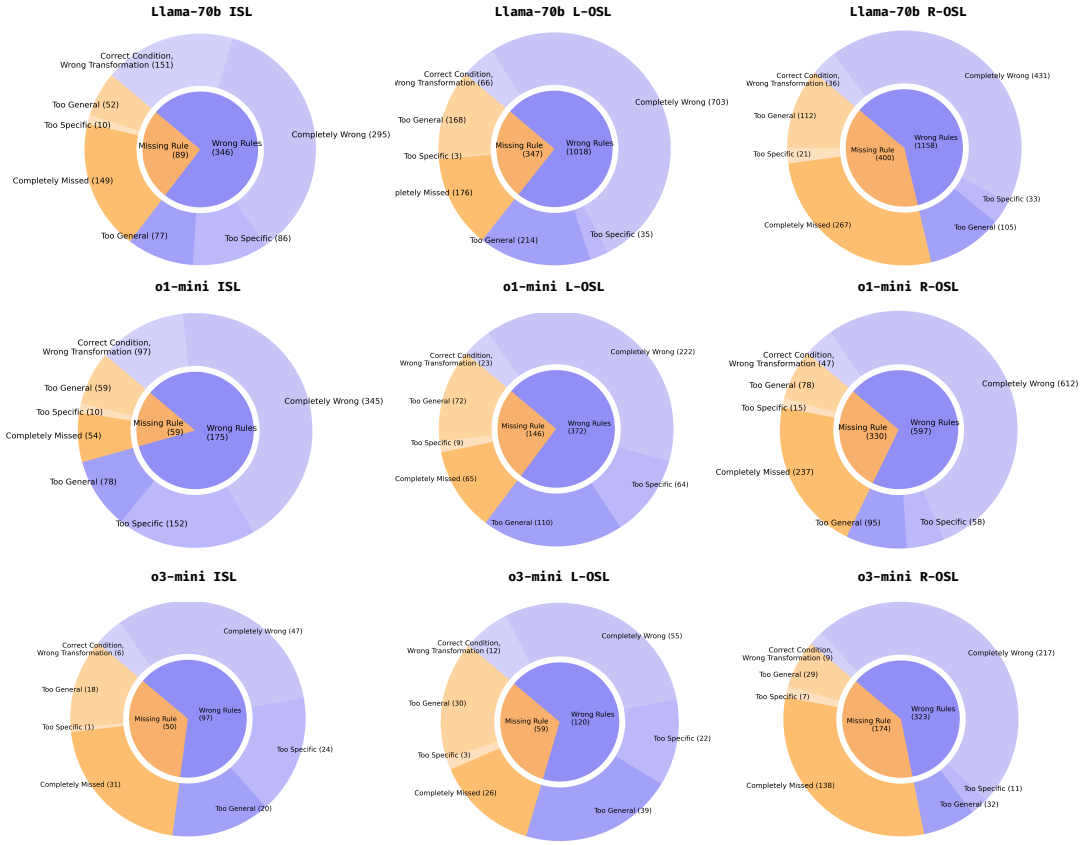

Figure 7: Error Type Analysis

C LEADERBOARD BASED ON INDUCTIONBENCH

To facilitate straightforward comparisons among different LLMs, we introduce a two-part benchmark leaderboard: a *standard leaderboard* and an *exploration leaderboard*. The *standard leaderboard* is based completely on the three function classes we talked about, and this leaderboard simply presents an aggregated score to directly reflect LLM's performance. The *exploration leaderboard* includes a slightly new design of function class and we will present the motivation and details below.

C.1 STANDARD LEADERBOARD

The standard leaderboard consists of 1,080 questions spanning three classes of deterministic regular functions: *ISL*, *L-OSL*, and *R-OSL* in equal proportion. Specifically, it includes:

- 360 ISL questions,
- 360 L-OSL questions,
- 360 R-OSL questions.

Within each function class, we have settings for $k \in \{2, 3, 4\}$, $|\Sigma| \in \{5, 6, 7, 8\}$, and number of rules $\in \{3, 4, 5\}$. Each unique parameter combination has 10 data points, totaling 360 points per function class. The performance metrics *recall*, *precision*, and *compatibility* are computed on a per-setting basis. We then form an overall *weighted average* to account for variations in function-space size:

Definition 8 *For a given setting characterized by $(k, |\Sigma|, r)$, the weight w is defined as $\frac{|\Sigma|^k}{\sum_{k'=2}^{k'=4}\sum_{s=5}^{s=8} s^{k'}}$, where k is the Markov window, $|\Sigma|$ is the alphabet size, and r is the minimal rule count.*

For each function class (ISL, L-OSL, R-OSL), we compute a weighted recall, precision, and compatibility according to the above scheme and then take the average of these three scores to produce the final leaderboard score for that class. The overall score across all three classes is the average of those class-wise scores.

Table 9 summarizes current leaderboard results for several representative models. Notably, even o3-mini achieves only a 5.69% compatibility score, largely because none of the models succeed on tasks where $k = 4$. Since those high-complexity settings receive substantially larger weights than cases where $k \in \{2, 3\}$, they disproportionately reduce the overall average.

model	average recall	average precision	average compatibility
Llama-3.1 8b	0.00	0.00	0.00
Qwen2.5-Coder-32B-Instruct	7.26	0.66	0.03
Llama-3.3-70b	6.55	5.76	0.12
DeepSeek-R1-Distill-Llama-70B	3.84	5.14	0.78
o3-mini	28.93	43.12	5.69

Table 9: Leaderboard Result

To balance the influence of different complexity settings, we additionally report an alternative evaluation metric that replaces each original weight with its logarithm. This approach dampens the dominance of $k = 4$ scenarios, yielding a more even distribution of weights across the benchmark's parameter space.

model	average recall	average precision	average compatibility
Llama-3.1 8b	0.00	0.00	0.00
Qwen2.5-Coder-32B-Instruct	7.48	6.60	0.48
Llama-3.3-70b	8.71	7.50	0.87
DeepSeek-R1-Distill-Llama-70B	23.17	24.66	8.63
o3-mini	57.58	63.89	33.93

Table 10: Leaderboard Result with Log Weight

C.2 Exploration Leaderboard

A key concern in using subregular function classes (*e.g.*, ISL, L-OSL, R-OSL) is that polynomial-time learning algorithms already exist for these classes, potentially allowing a trivial "hack" to achieve artificially high performance. Though we advocate not using the provbly correct algorithm for task solving so that we can genuinely evaluate LLM's inductive reasoning ability, to make sure, we introduce an *exploration leaderboard* that focuses on *Input-Output Strictly Local (IOSL)* functions: a more speculative class for which no known algorithm can reliably learn the entire function from finite data in finite time.

Rationale. Since IOSL lacks a proven polynomial-time learning procedure, successful performance here would more credibly reflect genuine inductive reasoning rather than the application of a known "shortcut" algorithm. Furthermore, IOSL functions have not been deeply studied in the literature, offering an opportunity to see whether LLMs can advance this open research area.

This is the definition of IOSL:

Definition 9 (IOSL) *A function f is IOSL if there is a k such that for all $u_1, u_2 \in \Sigma^*$, if $\textsc{Suff}^{k-1}(u_1) = \textsc{Suff}^{k-1}(u_2)$ and $\textsc{Suff}^{k-1}(f(u_1)) = \textsc{Suff}^{k-1}(f(u_2))$, then $\textsc{tails}_f(u_1) = \textsc{tails}_f(u_2)$.*

In essence, this condition requires the model to distinguish between input-based and output-based Markovian triggers, making the learned transformation highly non-trivial if no pre-existing algorithm is used.

Leaderboard Setup. The IOSL-based leaderboard contains 1,080 datapoints, mirroring the standard leaderboard in overall structure: $k \in \{2, 3, 4\}, |\Sigma| \in \{5, 6, 7, 8\}$, number of rules $\in \{3, 4, 5\}$. For each setting, there are 30 datapoints per setting (for equivalence to the standard leaderboard's size).

Since IOSL is not known to admit a finite-characteristic sample or minimal representation in the same sense as the deterministic classes, we introduce two adaptations for evaluation:

1. **Sample Size.** We arbitrarily fix the sample size at $2 * |\Sigma|^k$, as no characteristic sample is theoretically guaranteed.
2. **Evaluation Metrics.** We focus primarily on *compatibility*, as recall and precision hinge on the assumption of a unique minimal-length description, which may not exist for IOSL. If a model's generated rule set is compatible with the data, we then check whether its description length is shorter, identical, or longer than our function's reference length. A longer description indicates a definite failure to produce a minimal representation; shorter or equal does not guarantee minimality, but it at least suggests the model avoids obvious redundancy.

By presenting both a standard leaderboard (subregular classes with known learnability) and an exploration leaderboard (IOSL with no established finite-data algorithm), we offer a balanced view: models can demonstrate success in theoretically well-understood tasks while also exploring novel, under-constrained function classes—thereby reducing the concern that high performance might merely reflect an existing "hack."