A Comprehensive Evaluation of Inductive Reasoning Capabilities and Problem Solving in Large Language Models

Bowen Chen♦, Rune Sætre ♠, Yusuke Miyao♦
♦Department of Computer Science, The University of Tokyo
{bwchen, yusuke}@is.s.u-tokyo.ac.jp
♠ Computer Science, Norwegian University of Science and Technology (NTNU)
satre@ntnu.no

Abstract

Inductive reasoning is fundamental to both human and artificial intelligence. The inductive reasoning abilities of current Large Language Models (LLMs) are evaluated in this research. We argue that only considering induction of rules is too narrow and unrealistic, since inductive reasoning is usually mixed with other abilities, like rules application, results/rules validation, and updated information integration. We probed the LLMs with a set of designed symbolic tasks and found that even state-of-the-art (SotA) LLMs fail significantly, showing the inability of LLMs to perform these intuitively simple tasks. Furthermore, we found that perfect accuracy in a small-size problem does not guarantee the same accuracy in a larger-size version of the same problem, provoking the question of how we can assess the LLMs' actual problem-solving capabilities. We also argue that Chain-of-Thought prompts help the LLMs by decomposing the problem-solving process, but the LLMs still learn limitedly. Furthermore, we reveal that few-shot examples assist LLM generalization in out-of-domain (OOD) cases, albeit limited. The LLM starts to fail when the problem deviates from the provided few-shot examples.

1 Introduction

Recently, the development of LLMs has made great progress in various areas of artificial intelligence (AI), especially in Natural Language Processing (NLP). The performance of LLMs like GPT-3.5 (Brown et al., 2020) and GPT-4 (OpenAI, 2023) can even outperform humans on some professional tests, proving their ability to understand and solve complex natural language questions. One of the intriguing abilities of LLMs is reasoning, which is also one of the core abilities of human intelligence.

Reasoning, following this definition (Hurley, 2000), consists of deductive reasoning (Johnson-Laird, 2010), inductive reasoning (Hawthorne, 2021), and abductive reasoning (Douven, 2021). LLMs show surprisingly high performance on tasks requiring high-level reasoning ability, like programming (Xu et al., 2022) and mathematical problem solving (Imani et al., 2023). However, as the LLMs memorize the statistical word co-occurrences from the pre-training corpora containing such examples, it is hard to know the real reasoning ability of LLMs as they always generate specious answers. Therefore, evaluation at a fundamental level, e.g. symbolic level, is needed to accurately understand the reasoning abilities of LLMs.

This research focuses on inductive reasoning, which is the ability to derive common principles from finite observations. Recent inductive reasoning research in NLP (Yang et al., 2022; Li et al., 2023) focused mainly on rules induction from observations, but inductive reasoning in the real world is more complex than just rules induction.

As inductive reasoning is based on finite observations, which may contain only partial information, we cannot always expect the induced rules or results to be fully correct. Therefore, in the real world, under the surface of rules induction, the ability to validate induced rules/results and merge new rules with previous rules is equally important, and such ability to adapt to changing circumstances is important for building AI models suitable for real-world usage. To evaluate these abilities, we designed three symbolic tasks: 1) Grouping Polygons, 2) ordering named colors (Color Ordering), and 3) shifting characters in English text (Character Mapping).

We then define 3x5 experiments called Rules Application, Rules Induction, Results Validation, Rules Validation, and Rules Incorporation to evaluate the ability to apply rules, induce rules, validate induced results/rules, and merge new rules with previous rules, as depicted in Figure 1. We observe the LLMs failing on these tasks. Subsequent experiments explored the role of few-shot examples for generalization, the scalability of LLM perfor-

Findings of the Association for Computational Linguistics: EACL 2024, pages 323–339

mance with problem size, and the impact of the Chain-of-Thought prompts, namely:

1. For evaluated LLMs, the performance varies a lot between different experiments. This unstable LLM performance on symbolic inductive reasoning tasks is in contrast to their stable/robust performance on NLP tasks. Besides the instability, the task accuracy is low even for SotA LLMs, illustrating the weakness of LLMs in symbolic reasoning tasks.

2. In addition to low accuracy in Rules Induction and Rules Application, LLMs also perform poorly in Results/Rule Validation and Rules Incorporation. This suggests that besides focusing on the accuracy of LLMs, their ability to validate and check the generated results should be paid attention to.

3. LLMs can learn from few-shot examples and generalize beyond the given few-shot examples, but they still fail to learn scalable solutions from the examples, even when decomposing the problem-solving procedures through Chain-of-Thought (CoT) prompting.

4. While the LLMs may solve small-sized problems perfectly, the accuracy drops drastically when increasing the problem size. This provokes the question, "How can we prove that the LLM really holds the solution to solve specific types of problems?"

2 Related Research

2.1 Reasoning in LLMs

Reasoning is a core ability of human intelligence and an established research area in machine learning. Previously, even simple natural language reasoning tasks were very challenging for neural models (Santoro et al., 2018; Saxton et al., 2019).

However, the appearance of pre-trained language models like BERT (Devlin et al., 2019), with the commonsense knowledge encoded in the model through pre-training, largely improved the performance on NLP tasks, including reasoning tasks (Helwe et al., 2021). In recent years, with the scaling of model size, data size, and development of new architectures, different abilities have emerged from LLMs (Wei et al., 2022). Reasoning is one of those emerging abilities. Combining tricks like Chain-of-Thought (Wei et al., 2023) and In-Context Learning (Dong et al., 2023), the performance on natural language reasoning tasks is largely improved, even for tasks like mathematical reasoning (Lu et al., 2023), which was hard for neural models.

Evaluating LLMs on natural language reasoning tasks makes it difficult to know their reasoning abilities as they learn word co-occurrence relations from the pre-training corpus to aid in NLP reasoning tasks. To avoid the benefit of the encoded word/sentence/knowledge from pre-training and evaluate the reasoning ability at a more basic level, we create symbolic tasks to isolate semantic meaning to better evaluate LLMs' reasoning abilities.

2.2 LLM Probing

Probing is an important method to understand black-box neural networks with millions of parameters (Alain and Bengio, 2017). It is impossible to analyze them from a purely mathematical standpoint. Using probing tasks and analyzing the results gives us a peek hole to obtain insights into the inner mechanism of LLMs. Probing has proven to be an effective tool for analyzing the behavior of neural networks and their mechanisms since RNN-based networks (Nelson et al., 2020), Transformer-based Pre-trained Models (Johnson et al., 2020; Vulić et al., 2020), and then current, much larger LLMs (Kondo et al., 2023; Wei et al., 2023).

This study centers on symbolic task-based probing of LLMs. Recently, Anil et al. (2022) illustrated LLMs' limitations in tackling long-length problems in parity checking and variable assignment tasks. Additionally, Dziri et al. (2023) examined LLM's capabilities using computational graph-based symbolic tasks like logical grids and multiplication computation. Their findings show that LLMs solve tasks by breaking them into linearized subgraphs and matching each subgraph in the pre-trained corpus. The lack of genuine systematic problem-solving skills is evident when accuracy decreases as the graph depth increases.

Differing from previous research in symbolic probing, we do not aim at evaluating a single or specific ability, rather we set up different experiment configurations to evaluate multiple abilities centered around inductive reasoning.

3 Problem Formulation

3.1 Symbolic Tasks

We argue inductive reasoning requires various abilities. To evaluate those abilities, we designed three

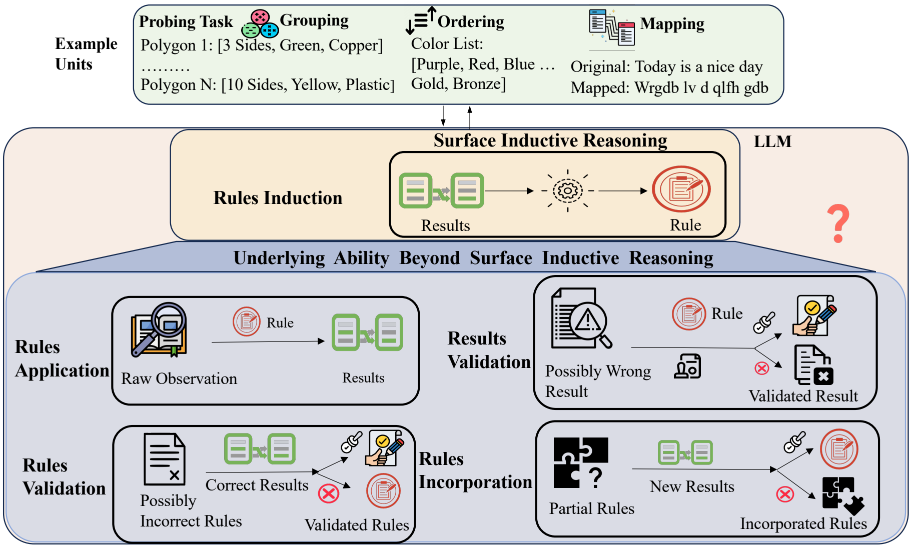

Figure 1: Evaluation Framework

symbolic tasks, explained in the following section.

Polygons Grouping In this task, we describe 30 polygons with different numbers of sides, colors, and material attributes. We also generate 15 grouping rules and the corresponding grouping results from following those rules.

Color Ordering In this task, we automatically generate a color priority dictionary with 20 colors in which a high-priority color should be given a high preference. We also generate corresponding sorted or unsorted color lists with 20 colors based on the color priority. Since we prompt both unsorted color list and color priority into the LLM, to prevent the LLM from just replicating the color priority list from the prompt to achieve a perfect sorted result, we remove five colors and duplicate five color units in the unordered color lists.

Character Mapping In this task, we form character mapping rules by mapping each English character to its three-index right-shifted counterpart, with a wrap-around between Z and A. We sample sentences from the App-Review (Grano et al., 2017) dataset with character lengths from 20 to 100 and mapped results following mapping rules.

3.2 Prompt Formulation

The prompt contains information about the target task to posit the LLM adapt to the task. Additionally, we may add few-shot question/prediction pairs for different tasks, named few-shot examples $F = \{f_1, f_2 \dots f_5\}$ to help the LLM respond with accurate answers. We use five examples for all few-shot experiments. Unless mentioned specifically, the prompt contents introduced below is the default prompt to the LLM in all tasks.

Task Illustration (T) The text prompt T sent to the LLM contains other necessary information consisting of four parts $T = \{T_d, T_i, T_f, T_r\}$. T_d is the Task Description with general information about the task. T_i is the Response Instruction, which states the LLM responses' expected content. T_f states the expected Response Format, and T_r is an optional Rules Text with the rules used in the tasks.

Units (S) Units S are the available symbolic units in a given task. The LLM L needs to know all symbolic units $S = \{s_1, s_2, \dots s_n\}$ prior to solving the corresponding symbolic task. For example, each polygon in the Grouping task is a unit.

Problem (X) After the Task Illustration and Units, we attach the problem text X to the prompt's end, and the LLM's prediction is denoted as $Y = \{y_1, y_2 \dots y_n\}$. The problem text, task illustration, and units differ based on task settings.

3.3 Scalable Solution (H)

As LLM solves the problem internally, we call such a hidden problem-solving procedure a solution, which is not a part of the prompt. In our task setting, we expect the LLM to have the Scalable Solution H that can be used to solve the prompted problem in any unit size. The Scalable Solution differs from Rules T_r. For example, in the Mapping

Role Positioning	You are a helpful assistant and you are supposed to follow the instructions that I give to you and perform the task as far as you can. Here we want to group different polygons into different groups based on their characteristics.
Problem Description	Problem Description ------ You will be given the possible attributes of different polygons with different Sides Numbers, Colors, and Materials. You will also be given the grouping rules that describe what types of polygons should be grouped together. You are supposed to group those polygons into different groups based on the grouping rules. Attributes ------ Sides Numbers: {SidesNumber} Colors: {Colors} Materials: {Materials}
Response Instruction	Response Instruction ------ Your final answer to this problem should contain the following information: 1. The polygons that belong to each group.
Response Format	Response Format ------ Following the Response Instruction, the format should be: Grouping Result: Group 0: Polygon x1, Polygon x2, ... Group 1: Polygon y1, Polygon y2, ... The above format is just an example, you should replace x1, x2, y1, y2 with actual polygons based on your analysis. The order of polygons in each group does not matter.
Few-Shots Examples (Optional)	Below are the grouping rules that describe what types of polygons should be grouped together: {First Example Rules} Now try your best to use those grouping rules for the group following polygons, your response should follow the Response Format. {First Example Polygons} Grouping Results: Group 0: Polygon 1, Polygon 2 …….
Prompted Problem	Below are the grouping rules that describe what types of polygons should be grouped together: {Rules} Now try your best to use those grouping rules for the group following polygons, your response should follow the Response Format. {Polygons}

Figure 2: Prompt Template for Rules Application Task of Polygon Grouping

task, the Scalable Solution is *mapping each character using its corresponding rules*, where mapping rules T_r serve as an input of the scalable solution.

3.4 Task Setting

We set up tasks to probe the LLM's inductive reasoning abilities in applying, inducing, validating, and rectifying results/rules, identifying new rules, and merging them with previous rules. Examples are shown in Table 1. Those tasks are designed on the principle that if the LLM holds the scalable solution H, these tasks are intuitively simple. The same solution can apply to every example, yielding perfect accuracy, as the scalable solution and the tasks remain constant regardless of unit size changes.

Rules Application In this task, we evaluate the ability to apply rules, and the problem text of this task is X_f. The LLM is asked to apply the given rules T_r to those symbolic units S and expect to obtain the correct results Y, formulated as:

$$L(T;S;X) = L(\{T_d, T_f, T_i, T_r\}; S; X_f) \xrightarrow{H} Y$$

Rules Induction In this task, we evaluate the ability to induce rules. We present the correct results Y obtained by applying the (hidden) rules to the given units. We denote the problem text for this task as X_l. We prompt the LLM to induce the (hidden) rules by observing the relation between units and the correct results, which can be formulated as:

$$L(T;S;X;Y) = L(\{T_d, T_f, T_i\}; S; X_l; Y) \xrightarrow{H} T_r$$

Results Validation In this task, we evaluate the ability to validate the results' correctness and correct the **results** if an error exists. The problem text of this task is X_r. We prompt the LLM with the rules and a (probably) wrong result $\hat{Y}$ with three errors generated randomly with 50% chance. The LLM is required to validate and/or correct the given result $\hat{Y}$. The LLM first answers whether the given result is correct. It is a binary classification problem denoted as $U_r = \{Yes, No\}$. If $U_r = Yes$, the LLM quits generation by outputting words like *None*. If $U_r = No$, the LLM applies rules to rectify the error and obtain new results $\overline{Y}$, formulated as:

$$\text{Let } L(\{T_d, T_f, T_i, T_r\}; S; X_r; \hat{Y}) \xrightarrow{H} U_r$$

$$\overline{Y} = \begin{cases} L(T; S; X_r; \hat{Y}; U_r) & \text{if } U_r = No \\ None & \text{if } U_r = Yes \end{cases}$$

Rules Validation In this task, we evaluate the ability to validate the correctness of the rules and correct the **rules** if errors exist and the problem text of this task is X_e. The prompted rules $\hat{T_r}$ are possibly wrong and may have three error rules generated randomly in 50% of the experiments, and the prompted correct result can help validate the correctness of the rules. Knowing whether the rules are correct is the first step for solving the problem, so we call that result $U_e = \{Yes, No\}$. If

	Task Illustration	Units	Problems		Predictions
Rules Induction	Inducing grouping rules through observing the grouping results.	Polygon 1: [3 Sides, Green, Copper], Polygon 2: [5 Sides, Red, Iron], ... Polygon N: [10 Sides, Yellow, Plastic]	Group 1:[Polygon 1, Polygon 3, . . .] . . . Group N:[Polygon 7, Polygon N, . . .]	Induce the grouping rules by observing the above results.	Induced Rules: Rule 1: 3 Sides, Green and Copper Rule 2: 5 Sides, Red and Iron ...
Rule Application	Applying grouping rules to given polygons to obtain the grouping results		Rule 1: 3 Sides, Green and Copper Rule 2: 5 Sides, Red and Iron ...	Apply the above grouping rules to the given polygons and give the results	Grouping Results: Group 1:[Polygon 1, Polygon 3, . . .] . . . Group N:[Polygon 7, Polygon N, . . .]
Results Validation	Validate the correctness of grouping results and rectify them if they are wrong		Rule 1: 3 Sides, Green and Copper ... Group 1:[Polygon 1, Polygon 2, . . .] Group 2:[Polygon 3, Polygon 6, . . .]...	Validate the correctness of the result first and rectify them if it is wrong	Correction Results Or Not: No Corrected Results: Group 1:[Polygon 1, Polygon 3, . . .] . . . Group N:[Polygon 7, Polygon N, . . .]
Rules Validation	Validate the correctness of rules and correct them if they are wrong		Rule 1: 3 Sides, Green and Copper ... Group 1:[Polygon 1, Polygon 3 . . .] Group 2:[Polygon 2, Polygon 6. . .]. . .	Validate the correctness of rules first and rectify them if it is wrong	Correction Rules Or Not: Yes Rules do not need correction
Rules Incorporation	Find whether new rules exist in the new results or not if so, induce new rules.		Rule 1: 3 Sides, Green and Copper . . . Group 1:[Polygon 1, Polygon 3, . . .] Group 2:[Polygon 2, Polygon 6, . . .]. . .	Find whether there exist new rules or not and induce them if necessary	New Rules Or Not: Yes New Inducted Rules: Rule 2: 5 Sides, Red and Iron . . .

Table 1: Different Task Examples in Polygons Grouping

$U_e = Yes$, the LLM finishes generation by outputting words like *None* as correct rules do not need correction. If $U_e = No$, the LLM corrects the wrong rules and obtain corrected rules $\overline{T_r}$ based on the correct results Y, which can be formulated as:

$$\text{Let } L(T = \{T_d, T_f, T_i, \hat{T_r}\}; S; X_e; Y) \xrightarrow{H} U_e$$
$$\overline{T_r} = \begin{cases} L(T; S; X_e; Y; U_e) & \text{if } U_e = No \\ None & \text{if } U_e = Yes \end{cases}$$

Rules Incorporation In this task, we evaluate the ability to identify new rules and merge new rules with previous rules if new rules exist where the problem text of this task is X_i. The prompted rules $\hat{T_r}$ and the results are correct, but the rules may be a part of the entire rule-set since we withhold three new rules in the given new result with a 50% chance. The LLM refers to the new result and identifies whether we can induce new rules from it or not. Identifying whether new rules exist is the first step, so we denote this binary classification results as $U_i = \{Yes, No\}$. If $U_i = No$, the LLM finishes generation with the word *None*. If $U_i = Yes$, the LLM should induce new rules $\ddot{T_r}$ based on the new given results Y, formulated as:

$$\text{Let } L(\{T_d, T_f, T_i, \hat{T_r}\}; S; X; Y) \xrightarrow{H} U_i$$
$$\ddot{T_r} = \begin{cases} L(T; S; X_i; Y; U_i) & \text{if } U_i = Yes \\ None & \text{if } U_i = No \end{cases}$$

We show an example of the prompt formulation in Figure 2 for the Rules Application Task for Polygon Grouping. As illustrated in the figure, the prompt first indicates the role of the LLM to posit the LLM in a position to solve the task. The following Problem Description contains the Task Illustration T and Units S which in this example is to group different polygons. Then Response Instruction tells how the model should respond so that the answer generated can be extracted easily. Then Few-Shot examples are optional depending on the experiment setting. Finally, the Prompted Problem contains the Problem X that the LLM should answer following all the information contained in the prompt. The content of each part changes with the different task settings, but all share the same backbone structure. [1]

4 Experiments[2]

In this study, all those tasks are automatically generated and can be automatically solved by the corresponding program as the solution for each problem is the same. Though humans may not solve the problem with perfect 100% accuracy due to humans making mistakes in following solution procedures like overlooking some rules, this does not mean humans cannot solve this problem as it is not caused by the inability of inductive reasoning. In the optimal situation, the performance for humans should be perfect as the program which is 100%.

4.1 Evaluated LLMs

Davinci (Brown et al., 2020) is a GPT3-based LLM trained with instruction tuning (Ouyang et al., 2022). We use the Text-Davinci-003 version[3] which has 175B parameters size.

GPT-3.5 (Brown et al., 2020) is one of the SotA LLMs currently. It is trained with both instruction tuning (Zhang et al., 2023) and RLHF, meaning reinforcement learning from human feedback (Christiano et al., 2023). Compared to Davinci, it is specially trained for chat purposes but still uses GPT-3 as a backbone structure.

[1] More details are in the Appendix A.3.

[2] Please refer to the Appendix for detailed settings of experiments and symbolic tasks.

[3] For brevity, Davinci is used to denote Text-Davinci-003.

Model	Task	Rules Application				Rules Induction			
		Zero-shot		Few-Shot		Zero-shot		Few-Shot	
		Par Acc	Full Acc	Par Acc	Full Acc	Par Acc	Full Acc	Par Acc	Full Acc
Davinci	Grouping	75.6	10.0	87.9	24.0	23.5	1.3	85.4	11.2
	Ordering	36.7	0.0	29.6	0.0	56.5	39.7	87.0	82.1
	Mapping	6.4	0.0	10.1	0.0	33.0	3.0	90.4	2.0
GPT-3.5	Grouping	88.5	23.7	90.6	33.4	88.6	24.2	91.4	24.4
	Ordering	32.9	0.0	35.5	0.0	54.5	46.1	93.6	88.9
	Mapping	33.5	6.3	39.9	10.1	68.4	6.8	89.0	8.0
GPT-4	Grouping	**99.5**	**95.3**	**99.9**	**98.8**	**95.5**	74.3	**99.9**	97.2
	Ordering	45.3	24.4	52.4	28.9	95.4	**96.6**	97.5	**98.8**
	Mapping	62.3	30.6	67.1	47.3	49.4	17.1	93.8	21.7

Table 2: Accuracy on Rules Application and Rules Induction. The best results for one LLM in different tasks in either Rules Application or Rules Induction are underlined, and the best results of all models are bold and underlined. Par Acc and Full Acc means Partial and Full Accuracy.

GPT-4 (OpenAI, 2023) is the current SotA LLM with a strong performance in various tasks. It even performs well on professional tests that require a high-level understanding of natural language.[4]

4.2 Evaluation Criteria

Validation Accuracy means the number of validation problems U that the LLM correctly predicts divided by the total number of examples.

Partial Accuracy means the percentage of sub-problems the LLM correctly predicted. It is only counted when sub-problems exist. For example, in the rule correction problem, $U_e = Yes$ means the prompted rules are correct, therefore the sub-problems do not exist so such an example is not counted into the calculation of Partial Accuracy.

Full Accuracy means the percentage that the LLM can correctly predict all sub-problems in a given problem. The Full Accuracy is only calculated for examples that have sub-problems.[5]

4.3 Results

4.3.1 Rules Application and Rules Induction

We discuss the Rules Application and Rules Induction together in Table 2 due to their contrasting nature that apply and induce rules and found:

1. For Rules Application, Grouping has the highest accuracy, followed by Mapping, then Ordering. For Mapping, applying mapping rules to text leads to unsemantic text, but LLMs are trained to generate meaningful text using Language Modelling, thus generating unsemantic mapped text is not straightforward. For the Ordering, the same color units exist in the unsorted list. The LLM needs to clarify and put the same colors together, but the LLMs struggle to find such a hidden procedure.

2. In Rules Induction, Ordering has the highest accuracy, followed by Grouping, then Mapping. In Ordering, the prompted ordered list equals directly telling the rules even with the deletion and repetition of some colors, leading to high accuracy. In Grouping, the LLM needs to check three polygon attributes to derive the rules, which lowers accuracy. In Mapping, the duplicated and mixed-case characters require the LLM to merge characters and induce case-insensitive rules. Such hidden steps make Mapping the most challenging task.

3. The accuracy for Rules Induction is lower than for Rules Application except for Ordering, which we have explained above, showing that Rules Induction is harder. GPT-4 performs better than GPT-3.5 and Davinci in both tasks, possibly due to a much larger pre-train size, instruction tuning size, and model size.

4. A high Partial Acc does not mean high Full Acc shows the prediction error scatters in each example rather than converging in several examples, meaning that the LLM tends to make small mistakes in each example.

[4]The evaluated Llama2 gives extremely low accuracy and we put its experiment results and analysis in the Appendix.

[5]We abbreviate Validation Accuracy, Partial Accuracy, and Full Accuracy as Valid Acc, Partial Acc, and Full Acc.

(a) Results Validation

Model	Task	Zero-Shot			Few-Shot		
		Valid Acc	Partial Acc	Full Acc	Valid Acc	Partial Acc	Full Acc
Davinci	Grouping	51.6	28.4	8.6	52.0	33.5	15.9
	Ordering	96.0	20.1	2.8	100	79.3	67.5
	Mapping	53.6	1.5	0.0	59.0	12.9	2.5
GPT-3.5	Grouping	55.6	27.3	10.9	66.0	28.2	11.0
	Ordering	96.0	32.6	9.3	100	53.3	25.9
	Mapping	50.3	0.1	0.0	50.3	5.4	0.9
GPT-4	Grouping	82.1	96.0	13.1	92.5	93.2	14.5
	Ordering	**100**	**98.9**	**93.7**	**100**	**98.7**	**95.6**
	Mapping	61.2	77.3	8.9	68.7	80.4	60.0

(b) Rules Validation

Model	Task	Zero-Shot			Few-Shot		
		Valid Acc	Partial Acc	Full Acc	Valid Acc	Partial Acc	Full Acc
Davinci	Grouping	50.8	51.8	7.8	46.5	66.5	19.3
	Ordering	57.4	82.1	2.4	68.2	77.4	39.2
	Mapping	50.3	16.2	11.8	51.8	59.7	42.0
GPT-3.5	Grouping	51.3	21.2	3.9	53.0	29.1	6.4
	Ordering	94.5	55.4	34.6	78.8	78.3	47.6
	Mapping	51.2	26.0	22.0	91.8	39.9	32.7
GPT-4	Grouping	67.6	**89.8**	52.2	90.8	93.5	59.4
	Ordering	**100**	86.5	**80.3**	**100**	**97.4**	**96.1**
	Mapping	50.7	82.2	54.6	84.2	95.5	94.3

Table 3: Model accuracy on Results Validation and Rules Validation. The best results for one LLM between different tasks are underlined, and the best results of all models are both bold and underlined.

4.3.2 Results Validation and Rules Validation

The Results Validation and Rules Validation are discussed concurrently due to their contrasting nature. The outcomes are presented in Table 3.

1. In Results Validation, Mapping has the lowest accuracy, followed by Grouping and Ordering. For Mapping, locating an error requires applying rules to the character at the corresponding index, requiring the LLM to count the sequence length and locate it, but LLMs struggle to do such precise manipulation. For Grouping, the LLM needs to check three attributes to locate the error, which is comparatively easier. For Ordering, identifying an error merely needs checking color units sequentially with the prompted color preference.

2. For Rules Validation, Grouping has the lowest accuracy, followed by Mapping and Ordering. For Grouping, LLM has to induce rules from grouping results first and compare them with the possible wrong rules, and such a hidden step increases the difficulty. For Mapping, just apply the rule to each original and mapped character to check if conflicts exist. It is relatively easier to locate and correct the error. For Ordering, similarly, an ordered color list is another representation of rules, making it easy to both validate and correct.

3. LLMs give a low Valid Acc in all tasks except Ordering for reasons explained above. As validation is a binary classification problem, such accuracy means LLMs struggle to validate the correctness of results even for GPT-4, even though GPT-4 scores are slightly better.

4. Rules Validation have a higher Partial and Full Acc than Results Validation. This is because the rule sizes are much smaller than the unit size and we have several rules but dozens of units, making Rule Validation easier due to the smaller prediction space.

5. In all LLMs, the few-shot can boost the accuracy in Partial Acc and Full Acc while the improvement in Valid Acc differs, showing that

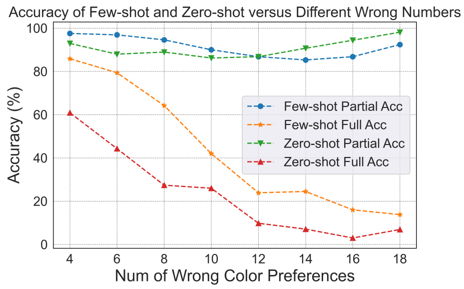

Figure 3: Accuracy Change in Few-Shot Generalization

the ability to learn to validate the results/rules from examples varies.

4.3.3 Rules Incorporation

The Rule Incorporation task can be considered a variant of Rules Induction where the LLM knows partial rules but may need to complete them based on whether the given results contain new rules. From the results in Table 4, we can see:

1. In the Zero-Shot setting, LLMs show no obvious preference regarding Valid Acc in either task, while Few-Shot improves it in the Ordering task, but Davinci and GPT-3.5 still fail to identify new rules from results. GPT-4 shows a high Valid Acc, meaning that the ability to validate new rules may be an emergent ability when LLMs reach a certain model size.

2. In contrast to Rules Induction, a decrease in Full Acc in Ordering and Grouping tasks is observed, which is counter-intuitive given the partial rules should enhance results as it reduces the prediction space for rules. This may be because even though the prediction space is narrowed, identifying new rules and merging them with existing rules poses another difficulty for LLMs. Conversely, the Mapping tasks benefit from given partial rules, which reveals that rules can be completed by right-shifting three indices, thereby simplifying the rule inference compared to other tasks.

4.3.4 Few-Shot Generalization

The task accuracy of LLMs can be largely improved by adding few-shot examples. However, this is when the few-shot examples are not out-of-distribution with the problem prompted. This leaves a question: *Does the LLM learn the scalable solution of the task or just fit into the answer pattern from few-shot examples?* We discuss this problem using GPT-4 and the Rules Validation of the Ordering tasks. We set the few-shot examples with three error color preferences, but the final problem includes more. We compare the zero-shot and few-shot settings results depicted in Figure 3:

1. The few-shot setting has higher accuracy than the zero-shot setting, proving that the LLM learns to generalize beyond the few-shot examples with three wrong color preferences. Notably, the few-shot setting initially exhibits an accuracy advantage exceeding 20%.

2. Providing few-shot examples does not make the LLM generalize to all situations as the accuracy decreases like in the zero-shot setting and even gets close to that accuracy in extreme situations, suggesting LLM only learns limitedly from few-shot examples.

3. The increasing trend in Partial Acc after 12 wrong preferences is because the random chance of picking out a wrong color preference increases with more wrong colors.

4.3.5 Increased Unit Size

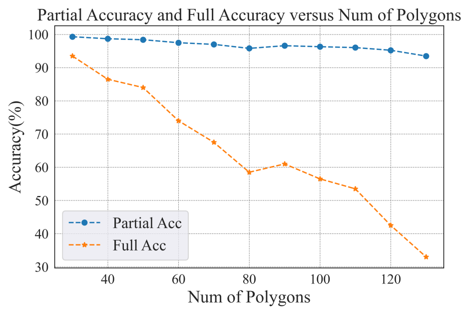

Figure 4: Accuracy Regarding Increased Polygons Size

Instead of increasing the task's difficulty, we evaluate the situation in which the underlying structure of the task remains fixed, but the unit size increases. GPT-4 has near-perfect Rules Application accuracy in the Polygon Grouping task, indicating it may hold a scalable solution for this. We want to see whether the performance remains stable when the unit sizes increase. The results in Figure 4 show the GPT-4's accuracy with increased polygon size:

1. The Full Acc decreases, showing the LLM cannot scale its performance with increased unit size even when small and larger problems share the same structure. This shows that the LLM does not hold the scalable solution despite its high accuracy in small-size problems.

Model	Task	Zero-shot			Few-Shot		
		Valid Acc	Partial Acc	Full Acc	Valid Acc	Partial Acc	Full Acc
Davinci	Grouping	51.0	30.1	3.9	52.0	37.2	5.8
	Ordering	49.2	37.9	0.5	87.5	26.1	3.4
	Mapping	49.3	49.0	5.6	49.3	87.4	34.5
GPT-3.5	Grouping	50.3	33.4	10.9	54.8	42.5	16.9
	Ordering	51.1	33.3	8.0	66.0	72.0	39.1
	Mapping	51.4	78.7	28.8	52.0	91.4	55.2
GPT-4	Grouping	99.7	**96.1**	**89.5**	99.5	**98.0**	**94.6**
	Ordering	**100**	96.2	82.8	**100**	96.1	89.9
	Mapping	95.1	94.4	56.0	97.2	95.9	62.3

Table 4: Model Accuracy on Incorporation. The best results for one LLM between different tasks are underlined and the best results of all models are both bold and underlined.

CoT Few-Shot Nums	Partial Acc	Full Acc
w/o CoT 5 Shot	52.4	28.9
CoT-1 Shot	82.6	58.6
CoT-2 Shot	83.0	57.7
CoT-3 Shot	84.6	55.8
CoT-4 Shot	84.9	62.2
CoT-5 Shot	**85.0**	**62.3**

Table 5: Chain-of-Thought Experiment

2. The Partial Acc is relatively stable, meaning the LLM predicts with stable accuracy for each sub-problem. However, the increased unit size enlarges the sub-problem size, which increases the expectation value of prediction error, naturally reducing the Full Acc.

4.3.6 Does Chain-of-Thought help?

In this experiment, we discuss to what extent the Chain-of-Thought (CoT) helps the LLM to solve the task. We evaluate GPT-4 in the Ordering of Rules Application task as even GPT-4 performs poorly in the few-shot setting. The CoT prompt shows the process of checking each color's preference and reordering the list based on acquired preferences. We reveal information on the scalable resolution to the LLM through those intermediate steps. From results in Table 5, we can see that:

1. From the results, the CoT-prompted model greatly improves the accuracy, leading to more than 35% accuracy gain in the 5-shot. This shows that the LLM learns to follow intermediate steps exposed by the CoT prompt, but it is still far from perfect accuracy, showing that a scalable solution is not learned.

2. We observe an inconsistency in accuracy improvement with increased few-shots. The accuracy decreases in the two or three-shot settings compared to one-shot, while the enhancement in the five-shot setting over one-shot is just 3.5%. This could be because each Color Ordering example has a different color preference and an unordered list (independent and not correlated with each other), so information from five examples is not substantially better than from just one.

5 Conclusion

In this research, through designed symbolic probing tasks, we probed the LLMs' abilities centered around inductive reasoning, including Rules Induction, Rules Application, Results/Rules Validation, and Rules Incorporation. We found that LLMs fail to correctly induce or apply rules in simple symbolic tasks and cannot or even fail to validate the correctness of results/rules or identify and merge new rules given new results. This suggests that not just improving prediction accuracy, but also making the LLM identify what is correct and wrong, and being able to identify new information from new examples are important.

Our experiments show that near-perfect accuracy in small-sized tasks does not imply that LLM performance scales well to a larger sized task. In this sense, it raises the question: *how can we prove that the LLM knows how to solve a problem/task?*

We also notice that few-shot examples help the model to generalize to unseen situations, but do not make the model able to solve the problem in all situations. Through the CoT-enhanced prompt, we see a significant performance improvement, stating that CoT helps the model to understand the scalable solution of a task in which the CoT prompt exposed more information about scalable solutions.

6 Limitations

We did not evaluate all available LLMs due to limited computational resources and service-restrictions (area limitation, wait-list, etc.). Instead, we selected several representative and strong LLMs that are easy to access. We are only able to run Llama2 models up to 13B, but we found that they do not even understand the prompt instructions correctly at those model sizes. This is despite following the correct way to prompt it, as described in the Llama2 paper (Touvron et al., 2023)[6] and in discussions[7] in the research community[8]. Please refer to Appendix A.1.2 for the results analysis of Llama2. Additionally, as probing research, our final goal was not actually to try to solve the symbolic tasks proposed in this paper, but that may be a separate goal in more powerful future research.

It is also possible that the accuracy can be further improved by using different prompts. We have tried various prompt designs and multiple prompts to make the LLMs give their best performance. The current prompt design gives the best accuracy among the prompts we have experimented with, though we do not deny that other prompts can improve the performance further. However, due to the number of possible prompts being infinite, we cannot exhaust them. We chose the best prompt among all the ones we have tried so far, and keep using it right now.

Additionally, all proposed symbolic tasks may be completely solvable if we prompt the LLM to use an external API like a sorting function or a pre-programmed function or even write its own code/program that can solve the given task. We argue that using such a tool to solve this problem is based on human-constructed knowledge, which equals making the human solve the task, not testing if the model can solve it. From a human perspective, those tasks are solvable even without external tools. Understanding rules, applying rules, discovering errors, and concluding on a general solution to a problem are fundamental aspects of intelligence that should be achieved even without external assistance from outside the model/brain.

[6] http://huggingface.co/blog/llama2#how-to-prompt-llama-2

[7] www.reddit.com/r/Localllama/comments/155po2p/get_Llama_2_prompt_format_right/

[8] http://twitter.com/osanseviero/status/1682391144263712768

7 Ethical Considerations

According to the terms-of-service of the OpenAI-provided API, its output (obtained data, model, etc.) cannot be used to compete with OpenAI.

We declare that we have no such intention of doing so. The purpose of this research is not to develop or produce any model or any data nor any method that aims to compete with OpenAI produced model, including the GPT3 (Text-Davinci) series, GPT-3.5 series, GPT-4 series, and all other OpenAI products (current or future improvements), released or coming models. We ask any follow-up researchers who cite this paper to also refrain from such competition in their follow-up research.

References

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning.

Igor Douven. 2021. Abduction. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, Summer 2021 edition. Metaphysics Research Lab, Stanford University.

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

Giovanni Grano, Andrea Di Sorbo, Francesco Mercaldo, Corrado A. Visaggio, Gerardo Canfora, and Sebastiano Panichella. 2017. Android apps and user feedback: A dataset for software evolution and quality improvement. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics*, WAMA 2017, page 8–11, New York, NY, USA. Association for Computing Machinery.

James Hawthorne. 2021. Inductive Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, Spring 2021 edition. Metaphysics Research Lab, Stanford University.

Chadi Helwe, Chloé Clavel, and Fabian M. Suchanek. 2021. Reasoning with transformer-based models: Deep learning, but shallow reasoning. In *3rd Conference on Automated Knowledge Base Construction*.

Patrick J. Hurley. 2000. *A Concise Introduction to Logic*. Wadsworth, Belmont, CA.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. MathPrompter: Mathematical reasoning using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 37–42, Toronto, Canada. Association for Computational Linguistics.

Devin Johnson, Denise Mak, Andrew Barker, and Lexi Loessberg-Zahl. 2020. Probing for multilingual numerical understanding in transformer-based language models. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 184–192, Online. Association for Computational Linguistics.

Phil Johnson-Laird. 2010. Deductive reasoning. *WIREs Cognitive Science*, 1(1):8–17.

Kazushi Kondo, Saku Sugawara, and Akiko Aizawa. 2023. Probing physical reasoning with counter-commonsense context. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 603–612, Toronto, Canada. Association for Computational Linguistics.

Yitian Li, Jidong Tian, Caoyun Fan, Wenqing Chen, Hao He, and Yaohui Jin. 2023. MTR: A dataset fusing inductive, deductive, and defeasible reasoning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10078–10089, Toronto, Canada. Association for Computational Linguistics.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning.

Max Nelson, Hossep Dolatian, Jonathan Rawski, and Brandon Prickett. 2020. Probing RNN encoder-decoder generalization of subregular functions using reduplication. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 167–178, New York, New York. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report.

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

Adam Santoro, Felix Hill, David G. T. Barrett, Ari S. Morcos, and Timothy P. Lillicrap. 2018. Measuring abstract reasoning in neural networks. *ArXiv*, abs/1807.04225.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7222–7240, Online. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy

Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Frank F. Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, MAPS 2022, page 1–10, New York, NY, USA. Association for Computing Machinery.

Zonglin Yang, Li Dong, Xinya Du, Hao Cheng, Erik Cambria, Xiaodong Liu, Jianfeng Gao, and Furu Wei. 2022. Language models as inductive reasoners.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2023. Instruction tuning for large language models: A survey.

A Appendix

A.1 Experiment Settings

A.1.1 Model Setting

For the Text-Davinci-003, we set the LLM to have zero temperature. For the GPT-3.5, we used the gpt-3.5-turbo-16k version. We set the temperature as 0 since we want to output for LLM to be stable, determinative, and reproducible. Additionally, we want the LLM to follow the instructions given in the prompt exactly. Setting the temperature is a good solution to make the LLM follow the instructions exactly.

For the GPT-4, we used the June 2023 version. Similarly, we also set the temperature as 0 to make the LLM produce deterministic results.

A.1.2 Llama2

For the Llama2-13b model, we show its experiment results in Table 6 and Table 7 and Table 8.

Regarding the results in Rules Application and Rules Induction, we can see that:

1. From the results in Table 6, we can see that Llama2 fails significantly in both the zero-shot and the few-shot settings. Especially in the Rules Application, the Llama2 gives zero Full Accuracy. Additionally, the Partial Accuracy is also low in Rules Application, even with few-shot examples showing that Llama2 may not be able to learn from those examples.
2. Regarding the Rules Induction, the accuracy is slightly better, even though it is far from satisfying. We can see that except for Ordering, in which the rules are easy to obtain from the prompted ordered color lists, the Llama2 also fails significantly in other tasks. For the Grouping and Mapping task, even with few-shot examples, the Full Accuracy is still only 2.9% and 2.0%.

Regarding the results in Results Validation and Rules Validation. From the results in Table 7.

1. Firstly, the Llama2 also fails to validate the correctness of results or rules in both the zero-shot setting and the few-shot setting.
2. Similarly, it also fails to correct the results. Even in the Gropuing with the few-shot setting, its performance is still just 8.9%. In other tasks, the performance is simply zero accuracy or close to zero accuracy.
3. In the Rules Validation, we have similar results. The Llama2 is also not able to validate the correctness of rules. Additionally, the Full Accuracy is also low.

Regarding the results of Rule Incorporation. From the results in Table 8, we can see:

1. The Llama2 also fails to identify new rules. This means that Llama2 cannot find new rules in the given results.
2. Additionally, the performance is also low in both the zero-shot setting and the few-shot setting.
3. The few-shot examples improve the Partial Accuracy a little, but do not improve the Full Accuracy.

We also did a brief case analysis of Llama2, and we found that in most cases, even following the desired response format to generate the answer is difficult. This means that it is hard to extract Llama2's prediction for the problem as it can be expressed in various ways even when we set its temperature parameter as zero, expecting it to follow the instructions. Additionally, Llama2 seems to repeat some tokens and also generate meaningless noise random tokens, which cannot be considered as an answer since it is meaningless.

Model	Task	Rules Application				Rules Induction			
		Zero-shot		Few-Shot		Zero-shot		Few-Shot	
		Par Acc	Full Acc	Par Acc	Full Acc	Par Acc	Full Acc	Par Acc	Full Acc
Llama2	Grouping	1.7	0.0	2.7	0.0	12.6	0.0	24.3	2.9
	Ordering	**34.4**	0.0	**35.4**	0.0	**38.9**	**29.7**	59.4	**40.4**
	Mapping	8.6	0.0	2.5	0.0	8.3	0.5	**89.8**	2.0

Table 6: Accuracy on Rules Application and Rules Induction. The best results are bold and underlined. Par Acc and Full Acc mean Partial and Full Accuracy respectively.

(a) Results Validation

Model	Task	Zero-Shot			Few-Shot		
		Valid Acc	Partial Acc	Full Acc	Valid Acc	Partial Acc	Full Acc
Llama2	Grouping	**58.0**	2.0	0.0	**58.0**	**23.5**	**8.9**
	Ordering	52.3	**3.4**	0.0	54.6	12.7	1.3
	Mapping	48.7	0.0	0.0	50.0	0.8	0.0

(b) Rules Validation

Model	Task	Zero-Shot			Few-Shot		
		Valid Acc	Partial Acc	Full Acc	Valid Acc	Partial Acc	Full Acc
Llama2	Grouping	49.3	**9.0**	0.0	50.2	**28.9**	1.3
	Ordering	49.3	8.4	0.0	48.4	0.0	0.0
	Mapping	**50.3**	8.3	3.4	**55.3**	15.8	**8.9**

Table 7: Model Accuracy on Results Validation and Rules Validation. The best results are both bold and underlined.

A.2 Task Setting

For the tasks evaluated in this research, we all randomly generated 500 examples for each task. For example, for the Character Mapping task, we randomly sample 500 sentences from datasets with character lengths from 20 to 100. The number of examples is also the same for other tasks. To notice that we have made sure that the possible combination of units is much larger than 500 examples so that it is not likely that we may generate the same examples twice in an experiment. Additionally, we use five random seeds [714, 123, 889, 912, 743], and the results are averaged over 5 random seeds. By fixing random seeds. we can make sure that each run produces the same generation of units, the errors in rules or results, and the new rules in the new given results.

A.2.1 Polygon Grouping Setting

We generate 30 polygons for each input example. Those polygons are randomly generated from the provided color list, sides number list, and material list.

The sides number list is [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

The color list is ['red', 'blue', "while", "black", "yellow", "purple", "gray", "cyan", "brown", "indigo"]

The material list is ['metal', 'plastic', "glass", "sliver", "gold", "copper", "bronze", "diamond", "jade"]

A polygon is generated through sampling from each attribute.

A.2.2 Character Mapping Setting

The text is chosen from the aforementioned App-Review dataset. We filter out sentences with character lengths either longer than 100 characters or shorter than 20 characters. Based on such conditions, we sample 500 examples from the filtered dataset as the data to be evaluated.

A.2.3 Color Ordering Setting

The color list that is used in this research contains the following colors ['Red', 'Blue', 'Green', 'Yellow', 'Orange', 'Purple', 'Pink', 'Brown', 'Black', 'White', 'Gray', 'Silver', 'Gold', 'Indigo', 'Turquoise', 'Cyan', 'Magenta', 'Lavender', 'Maroon', 'Beige', 'Teal', 'Navy', 'Olive', 'Coral', 'Salmon', 'Peach', 'Ivory', 'Tan', 'Lilac', 'Skyblue', 'Mint', 'Slate', 'Turmeric', 'Ruby', 'Emerald', 'Tangerine', 'Pewter', 'Champagne', 'Mauve', 'Brick', 'Forest', 'Mustard', 'Chocolate', 'Sapphire', 'Blush', 'Ash', 'Coral', 'Steel', 'Apricot', 'Pearl']. Each time, we randomly sample 20 colors from the whole list and randomly rank each color in the list to form the color preference dictionary. When prompting the LLM to induce rules based on correct output examples, we partition long color lists into several sub-lists to prevent the model from directly copying the given results to obtain the correct color preference without reasoning. The LLM should be able to merge those lists to produce the whole color preferences list.

A.3 Prompt Examples

As illustrated in the examples, first, we prompt the Role of the LLM to posit its general target of the task. Then, we prompt with a more detailed explanation of the tasks and provide detailed information about what the task is. After the Problem Description, we write the Response Instruction, which illustrates what answer we expect the model to respond with. We also added the Response Format to the LLM to make it generate the content following that format to let us extract the answers easily by parsing the output of the LLM. Then, depending on the task setting, we may attach the optional Few-Shots Examples after the Response Format. Notice that any content that is closed by the "" bracket pair is a placeholder. It will be replaced by the actual answer or prompted units. For example, "First Example Rules" means that this is the first example among few-shots examples. In the actual prompt, it will be replaced by actual rules. Also, the same for the "First Example Polygons", in which we will prompt the model with actual polygons that the model needs to group using the rules. Finally, after the Few-Shots Examples, we attach the actual prompted problem to the model with corresponding rules and polygons by replacing "Rules" and "Polygons" with the actual initiated rules and polygons for the problem.

We also show the prompt of other symbolic tasks. The prompts used for Character Mapping in all tasks are in Figure 5 and 6, which shows the prompt for Rules Application, Rules Induction, Results Validation, Rules Validation, and Rules Incorporation, respectively.

The prompts used for Polygons Grouping in all tasks are in Figure 7 and 8, which shows the prompt for Rules Application, Rules Induction, Results Validation, Rules Validation, and Rules Incorporation, respectively.

The prompts used for Color Ordering in all tasks are in Figure 9 and 10, which shows the prompt for Rules Application, Rules Induction, Results Validation, Rules Validation, and Rules Incorporation, respectively.

Model	Task	Zero-shot			Few-Shot		
		Valid Acc	Partial Acc	Full Acc	Valid Acc	Partial Acc	Full Acc
Llama2	Grouping	48.0	2.7	0.0	50.6	13.4	0.0
	Ordering	42.1	4.4	0.0	42.1	4.6	0.0
	Mapping	**51.3**	**8.5**	0.0	**51.3**	**21.1**	**7.7**

Table 8: Model Accuracy on Incorporation. The best results for one LLM between different tasks are underlined, and the best results of all models are both bold and underlined.

Mapping Rule Application

```
You are a helpful assistant, and you are supposed to follow the instructions that I give to you and perform the task as far as you can. Here, we want to transform the source text to the altered text by following the rules given below.
Problem Description
------
You will be given a set of rules that maps an English character to another character. You are supposed to follow the rules and transform the source text into the altered text.

Rules
------
Below are the character mapping rules that map each English character to another English character. Those rules work for both Uppercase and Lowercase:
{Rules}

Response Instruction
------
Your final answer to this problem should contain the following information:
1. The text that is mapped.

Response Format
------
Following the Response Instruction, the format should be:
Result:
Altered: MappedText
The above MappedText is just a variable which is the text that is mapped from the original text. Replacing it with the text that is mapped.

Question
------
Now try your best to map the Original text to the Altered text using the above rules and Response Format:
Original: {Original}

Remember your response must follow the response format.
```

Mapping Rules Induction

```
You are an inductive reasoner, and you can induct rules from examples correctly. You are given pairs of source text and altered text, and you are supposed to find the rules that map each English character to another.

Problem Description
------
You are given a set of pairs of source text and altered text, and you are supposed to find out the rules that map each English character in the source text to the corresponding altered text. You should ignore the non-English characters like space, numbers, question marks, etc. You should also ignore the case of the English character, which means you should treat the uppercase and lowercase as the same character.

Response Instruction
------
Your final answer to this problem should contain the following information:
1. The rules are used to map each English character to another.
2. Do not produce redundant rules, which means if there are two rules that map the same character to the same character, you should only respond to one of them.
3. The mapping character should be an uppercase English character.

Response Format
------
Rules:
Original: x1 -> Altered: y1
Original: x2 -> Altered: y2
……
Above Sides x1, y1, x2, and y2 are just variables, replacing them with English characters, which should be only uppercase English characters.

Question
------
Now try your best to induct the mapping rules from the following Original and Altered pair:
Original: {Original}
Altered: {Altered}
Remember your response should follow the response format.
```

Mapping Results Validation

```
You are an accurate error-checking assistant, and you can identify errors correctly. You have access to several pre-defined rules that map each English character to another English character, and you are given an Original and Altered text pair. The Altered string is obtained by mapping each English character in the Original text one by one using those pre-defined rules. However, the mapping for each character may not be correct. You are supposed to find out whether the mapping from the Original to the Altered text is correct or not. If not, locate the position of the error character and rectify it.

Problem Description
------
You are given a set of rules that maps each English character to another English character. You are supposed to check whether we can get the altered text from the original text using those rules. If not, where is wrong and locate the error.

Rules
------
Below are the rules that map each English character to another English character. Those rules work for both Uppercase and Lowercase:
{Rules}

Notice: Those rules only work for the English alphabet, and if you encounter non-English characters like space, numbers, question marks, etc, you don't have to check it.

Response Instruction
------
Your final answer to this problem should contain the following information:
1. Whether we can obtain the altered text by following the rules given above.
2. If the result is invalid, respond with the rectified Altered result.

Response Format
------
Following the Response Instruction, the format should be:
Validation Result:
Valid or Invalid
Rectified Results:
1. If the result is Valid, you respond with There is no character to correct.
2. If the result is Invalid, you respond with the rectified altered text in the following format.
Altered: RectifiedAlteredText
The above RectifiedAlteredText is just a variable, and you should replace it with the actual rectified altered text.

Question
------
Now try your best to answer the question for the following Original and Altered pair:
Original: {Original}
Altered: {Altered}
Remember your response should follow the response format.
```

Figure 5: Prompt Template for Mapping in Rules Application, Rules Induction and Results Validation

Mapping Rules Validation

```
You are an error rectifier. You have access to several pre-defined rules that map each English character to another English character, and you are given an Original and Altered text pair. The Altered string is obtained by mapping each English character in the Original text one by one using those pre-defined rules.

Problem Description
------
Some problems may happen to those rules due to some unexpected reasons; some of those rules may be disturbed, so the rules may not be correct anymore. You are supposed to rectify those rules by observing the Original and Altered string pair, where the Altered string pair is mapped by previous undisturbed correct rules. By checking the rules and the Original and Altered string pair, you can identify whether the given rules are correct or not.

Response Instruction
------
Your final answer to this problem should contain the following information:
1. Are the rules correct or not?
2. If not correct, what is/are the rectified one/ones.
3. If there are examples provided, follow the procedure for how examples solve the problem.

Response Format
------
Correct Rules or Not:
Yes or No
Rectified Rules:
1. If the result is Yes, you should respond with there is no rule to correct.
2. If the result is No, you should respond to the rectified rule/rules.
For example: Original: x1 -> Altered: y1
Here, x1 and y1 are just variables that represent English characters and do not have actual meanings; you should replace them with actual English characters based on your analysis.

Question
------
Try your best to answer the question using the above Response Format to determine whether the following rules contain incorrect rules or not:
{Rules}

Following is the correct ordered list of colors:
Original: {Original}
Altered: {Altered}

Now, you need to induct whether there are wrong rules existing in the given pre-defined mapping rules, and your response should follow the response format.
```

Mapping Rules Incorporation

```
You are an inductive reasoner. You have access to several pre-defined rules that map each English character to another English character, and you are given an Original and Altered text pair. The Altered string is obtained by mapping each English character in the Original text one by one using those pre-defined rules.

Problem Description
------
We have derived several rules based on previous Original and Altered pairs observations. Now, we have new data, and the problem is whether the new data can provide new rules or not. You are supposed to analyze whether the new pair can provide additional information or not. If the new pair presents new mapping rules, you should be able to identify them and incorporate them into the rules.

Response Instruction
------
Your final answer to this problem should contain the following information:
1. Does the Original and Altered pair provide new information or not
2. If it provides new information, what new rule can be inducted?
3. If there are examples provided, you should try to follow the procedure of how examples solve the problem.

Response Format
------
New Information Contained:
Yes or No
New Rules Inducted:
1. If you answer No in New Information Contained, you should respond with No.
2. If the answer Yes in New Information Contained, respond with the new inducted rules in the following format.
For example:
Original: x1 -> Altered: y1
Here, x1 and y1 are just variables that represent English characters and do not have actual meanings, and you should replace them with actual English characters based on your analysis.

Question
------
You have access to the following rules:
{Rules}

You have access to the following Original and Altered pairs:
Original: {Original}
Altered: {Altered}

Now, you need to check whether we can induct new rules from the given Original and Altered pair.
Remember your response should follow the response format.
```

Figure 6: Prompt Template for Mapping in Rules Validation and Rules Incorporation

Grouping Rules Application

```
You are a helpful assistant, and you are supposed to follow the instructions that I give to you and perform the task as far as you can. Here, we want to group different polygons into different groups based on their characteristics.

Problem Description
------
You will be given the possible attributes of different polygons with different Sides Numbers, Colors, and Materials. You will also be given the grouping rules that describe what types of polygons should be grouped together. You are supposed to group those polygons into different groups based on the grouping rules.

Attributes
------
Sides Numbers: {SidesNumber}
Colors: {Colors}
Materials: {Materials}

Response Instruction
------
Your final answer to this problem should contain the following information:
1. The polygons that belong to each group.

Response Format
------
Following the Response Instruction, the format should be:
Grouping Result:
Group 0:  Polygon x1, Polygon x2, …
Group 1:  Polygon y1, Polygon y2, …
…
The above format is just an example, and you should replace x1, x2, y1, and y2 with actual polygons based on your analysis. The order of polygons in each group does not matter.

Question
------
Below are the grouping rules that describe what types of polygons should be grouped together:
{Rules}

Now try your best to use those grouping rules to group the following polygons. Your response should follow the Response Format.
{Polygons}
```

Grouping Rules Induction

```
You are an inductive reasoner, and you can induct rules from examples correctly. You are given several polygons with different attributes like the Number of Sides, Colors, and Materials of Polygons. Additionally, you will be given a grouping result that those polygons are classified into different groups. You are supposed to find the grouping rules.

Problem Description
------
We first let you know the possible attributes for those polygons. Each polygon is a combination of those attributes. Then, we give you all the polygons that might be used for this problem. Now you have all the attributes and all the polygons, we give you the grouping results, and you are supposed to find the grouping rules that can be applied to those polygons to obtain the grouping results.

Attributes
------
Sides Numbers: {SidesNumber}
Colors: {Colors}
Materials: {Materials}

Response Instruction
------
Your final answer to this problem should contain the following information:
1. The rules that are used to group those polygons.

Response Format
------
Grouping Rules:
1. Polygons with x Sides, y Color, and z should be grouped together.
……
Above Sides x, y, and z are just variables, replacing them with actual numbers, color, and materials when producing the answer


Question
------
You have access to the following polygons:
{Polygons}

These are the grouping results for the above polygons with those attributes:
{GroupingResult}

Now, you need to induct the grouping rules following the above Problem Description, Response Instruction, and Response Format.
```

Grouping Results Validation

```
You are an accurate error-checking assistant, and you can identify errors correctly. You have access to several pre-defined rules that illustrate the grouping rules that you can use to group different polygons into different groups. You will be given grouping results and grouping rules and polygons. However, the grouping may not be correct. You are supposed to find out whether the grouping results are correct or not. If not, locate the error and rectify it.
Problem Description
------
You are given a set of grouping results of different polygons. You know the grouping rules and information about all the polygons. However, the grouping results may not be correct. You are supposed to find out whether the grouping results are correct or not. If it is incorrect, you should be able to locate and rectify the error.

Attributes
------
Below are all the attribute options for a polygon to have:
Sides Numbers: {SidesNumber}
Colors: {Colors}
Materials: {Materials}

Response Instruction
------
Your final answer to this problem should contain the following information:
1. Is the grouping result of polygons correct or not.
2. If the grouping results are not correct, give the rectified grouping result.

Response Format
------
Following the Response Instruction, the format should be:
Validation Result:
Correct or Incorrect
Rectified Results:
1. If the result is Correct, you respond with None.
2. If the result is Invalid, you respond with a new grouping result.
Group x: Polygon z_1, Polygon z_2, Polygon z_3..
Group y: Polygon n_1, Polygon n_2, Polygon n_3..
…
Above Group x, y, z_x and n_x are just variables, replacing it with actual group name when producing an answer

Question
------
You have access to the following polygons:
{Polygons}

Below are the rules that are used to group different polygons into different groups:
{Rules}

These are the grouping results for the above polygons with those attributes following the above rules:
{GroupingResult}

Now you need to check whether the grouping results are correct or not based on given polygons and rules. If not, give the rectified results, and your response must follow the response format.
```

Figure 7: Prompt Template for Grouping in Rules Application, Rules Induction and Results Validation

Grouping Rules Validation

```
You are an error rectifier. You have access to several pre-defined rules that illustrate the grouping rules you can use to group different polygons into different groups. You will be given the grouping results and grouping rules and polygons. However, the grouping rules may not be correct. You are supposed to find out whether we can obtain the grouping results following the grouping rules. If not, locate the error of the rules and rectify it.

Problem Description
------
Some problems may happen to group rules due to some unexpected reasons. Some of those rules may be disturbed, so the rules may not be fully correct. You are supposed to rectify those rules by observing the grouping results of polygons. By checking the rules and the grouping results, you can identify whether or not the given rules are correct.

Attributes
------
Below are all the attribute options for a polygon to have:
Sides Numbers: {SidesNumber}
Colors: {Colors}
Materials: {Materials}

Response Instruction
------
Your final answer to this problem should contain the following information:
1. Are the rules correct or not.
2. If not correct, what is/are the rectified one/ones.

Response Format
------
Following the Response Instruction, the response format is as follows:
Correct Rules or Not:
Yes or No
Rectified Rules:
1. If the result is Yes, you should respond with “There is no rule to correct”.
2. If the result is No, you should respond with the rectified rule/rules in the following format:
1. The wrong rule: x1 Sides, y1 Color, and z1 -> The correct rule: x2 Sides, y2 Color, and z2
….
The above x1 and x2, y1 and y2, z1 and z2 are just variables. You should replace it with the actual number of sides, colors, and materials. Remember, the wrong rule and the correct rule should be separated by “->” and are in one line.
Especially, x1, y1, and z1 are the variables for wrong sides, color, and material, and x2, y2, and z2 are the variables for correct sides, color, and material.

Question
------
Below are the polygons for this example:
{Polygons}

Below are the rules that are used to group different polygons into different groups, which may be incorrect:
{Rules}

These are the correct grouping results for the above polygons with those attributes:
{GroupingResult}
Now, you need to check whether the grouping rules are correct or not. If not, give the rectified results, and your response must follow the response format.
```

Grouping Rules Incorporation

```
You are an inductive reasoner. You have access to several pre-defined rules that hat illustrate the grouping rules that group different polygons into different groups. You will be given the grouping results, grouping rules, polygons, and available attributes of polygons. However, the grouping rules may not be complete. You are supposed to find out whether we can obtain new grouping rule/rules from the grouping results. If yes, discover new rules.

Problem Description
------
We have derived several rules based on previous observations of the grouping results of polygons. Now, we have new data, and the problem is whether the new data can provide new rules or not. You are supposed to analyze whether the new grouping results can provide additional information or not. If the new grouping results present new grouping rules, you should be able to identify them and incorporate them into the current rules.

Attributes
------
Below are all the attribute options for a polygon to have:
Sides Numbers: {SidesNumber}
Colors: {Colors}
Materials: {Materials}

Response Instruction
------
Your final answer to this problem should contain the following information:
1. Do the grouping results provide new information or not?
2. If given grouping results provide new rule/rules, what is/are the new rule/rules that can be inducted from the grouping results?

Response Format
------
Following the Response Instruction, your response should follow the following format:
New Rules or Not:
Yes or No
Added Rules:
1. If the above result is No, you should respond with “There is no rule to add” after Added Rules.
2. If the above result is Yes, you should respond  with the added rule/rules after Added Rules in the following format:
1. Polygons with x Sides, y Color, and z should be grouped together.
......
Above x, y, and z are just variables, replacing them with actual numbers, colors, and materials when producing answers.

Question
------
Below are the polygons for this example:
{Polygons}

Below are the rules that are used to group different polygons into different groups, which may be incomplete:
{Rules}

Below are the grouping results for the above polygons with those attributes:
{GroupingResult}

Now you need to check whether the grouping results provide new rules or not and your response must follow the response format.
```

Figure 8: Prompt Template for Grouping in Rules Validation and Rules Incorporation

Ordering Rules Application

You are a helpful assistant, and you are supposed to follow the instructions that I give to you and perform the task as far as you can. Here, we want to sort the given color lists that follow certain color preferences.

Problem Description

You will be given a set of rules that presents the color preferences. You will be given an unordered color list, and you should output the ordered list following the color preferences.

Color Set

You have access to the following colors:
{colors}

Response Instruction

Your final answer to this problem should contain the following information:
1. The resorted color list that is based on the given color preferences and unordered color list.

Response Format

Following the Response Instruction, the format should be:
Sorted Color List:
1. Color_1.
2. Color_2.
3. Color_3.
…
The above Color_x is just a variable here that does not hold any actual meaning. You should replace Color_x with actual colors from the given data.

Question

You have access to the following color preference rules that describe the correct color preference rank that you can use to sort the following unordered color list but do not output the color preference rank directly, and you should sort the following color list according to the following color preference rules:
{color_preference}

Now try your best to sort the following unordered color list according to the given color preference rules above, and your response should follow the response format. Don't just copy the color preference rank above, but try to sort the following color list according to the given color preference rules above:
{UnOrderedLists}

Ordering Rules Induction

You are an inductive reasoner, and you can induct rules from examples correctly. You are given an ordering result that the elements of the ordered result are different colors. You are supposed to find out the preference of the different colors, which means what color has the highest rank.

Problem Description

You are given an ordered list where, instead of ordering the numbers, the elements of the ordered list are colors. Through analyzing the unordered list and the ordered list, you are required to find out the rank of different colors.

Color Set

You have access to the following colors:
{colors}

Response Instruction

Your final answer to this problem should contain the following information:
1. The analyzed ranks of different colors.

Response Format

Color Ranking:
1. Rank 1 Color_1
2. Rank 2 Color_2
3. Rank 3 Color_3
4. Rank 4 Color_4
……
Color_x above is just a variable here that does not hold any actual meaning. You should replace Color_x with actual colors from the given data.

Question

Now try your best to induct the mapping rules from the following Original and Altered pair:
Original: {Original}
Altered: {Altered}

Remember your response should follow the response format.

Ordering Results Validation

You are an accurate error-checking assistant, and you can identify errors correctly. You have access to pre-defined ordering rules that show the preference of colors, and you are given ordering results based on those pre-defined ordering rules. However, the grouping results may not be correct. You are supposed to find out whether the given ordering results are correct or not. If not, you should be able to identify the errors and correct them.

Problem Description

You are given pre-defined ordering preferences that show the preference of colors, and you are given ordering results based on those pre-defined ordering rules. However, the grouping results may not be correct. You are supposed to find out whether the given ordering results are correct or not. If not, you should be able to identify the errors and correct them.

Color Set

You have access to the following colors:
{colors}

Response Instruction

Your final answer to this problem should contain the following information:
1. Whether the given ordering results are correct or not.
2. If not, what is/are the error/errors and the rectified one/ones.

Response Format

Correct Results or Not:
Yes or No
Rectified Results:
1. If the result is Yes, you should respond with "There is no error to correct".
2. If the result is No, you should respond with the rectified pair of colors in the following format, which means the color with the wrong priority (left-hand side) should be replaced with the right-hand side color.
For example:
The correct ordering results are:
Wrong Priority Color: Color_x -> Rectified Priority Color: Color_y
......
Color_x and Color_y above is just a variable that does not hold any meaning. You should replace Color_x with actual colors from the given data.

Question

You have access to the following color preference rules that describe the correct color preference:

{color_preference}

You have the following Ordered Color results that may not be correct:

{OrderedLists}

Now you need to induct whether the ordered colors follows the color preference rules or not and your response should follow the response format.

Figure 9: Prompt Template for Ordering in Rules Application, Rules Induction and Results Validation

Ordering Rules Validation

You are an accurate error-checking assistant, and you can identify errors correctly. You have access to pre-defined ordering rules that show the preference of colors, and you are given ordering results based on those pre-defined ordering rules. However, the color preference rules may not be correct. You are supposed to find out whether the given preference rules are correct or not. If not, you should be able to identify the errors and correct them.

Problem Description

You have access to pre-defined ordering rules that show the preference of colors, and you are given ordering results based on those pre-defined ordering rules. However, the color preference rules may not be correct. You are supposed to find out whether the given preference rules are correct or not. If not, you should be able to identify the errors and correct them.

Color Set

You have access to the following colors:
{colors}

Response Instruction

Your final answer to this problem should contain the following information:
1.Whether the given ordering rules are correct or not.
2.If not, what is/are the error/errors and the rectified one/ones.

Response Format

Correct Rules or Not:
Yes or No
Rectified Rules:
1. If the result is Yes, you should response with "There is no error to correct" following the Rectified Rules.
2. If the result is No, you should respond the rectified rule/rules only of the error rules in the following format.
Rank a : Color_x.
Rank b : Color_y.
Rank c : Color_z.
……
Color_x, Color_y, Color_z above is just a variable here that does not hold any actual meaning. You should replace them with actual colors from the given color set.
The a, b, and c after the RANK represent the rectified rank of the color. You should replace them with actual correct rank of the color.

Question

You have access to the following color preference rules that may contain incorrect rules:

{color_preference}

Following is the correct ordered list of colors:

{OrderedLists}

Now you need to induct whether there are wrong rules existing in the given pre-defined color preference rules and your response should follow the response format.

Ordering Rules Incorporation

You are an inductive reasoner. You have access to several pre-defined rules that hat illustrate the grouping rules that group different polygons into different groups. You will be given the grouping results, grouping rules, polygons, and available attributes of polygons. However, the grouping rules may not be complete. You are supposed to find out whether we can obtain new grouping rule/rules from the grouping results. If yes, discover new rules.

Problem Description

We have derived several rules based on previous observations of the grouping results of polygons. Now, we have new data, and the problem is whether the new data can provide new rules or not. You are supposed to analyze whether the new grouping results can provide additional information or not. If the new grouping results present new grouping rules, you should be able to identify it and incorporate them into the current rules.

Attributes

Below are all attribute options for a polygon to have:
Sides Numbers: {SidesNumber}
Colors: {Colors}
Materials: {Materials}

Response Instruction

Your final answer to this problem should contain the following information:
1. Do the grouping results provide new information or not?
2. If given grouping results provide new rule/rules, what is/are the new rule/rules that can be inducted from the grouping results?

Response Format

Following the Response Instruction, your response should follow the following format:
New Rules or Not:
Yes or No
Added Rules:
1. If the above result is No, you should respond with "There is no rule to add" after Added Rules.
2. If the above result is Yes, you should respond with the added rule/rules after Added Rules in the following format:
1. Polygons with x Sides, y Color, and z should be grouped together.
……
Above x, y, and z are just variables, replacing them with actual numbers, colors, and materials when producing answers.

Question

You have access to the following original color preference rules that may be incomplete:

{color_preference}

Following is the ordered list of colors that may provide new information:

{OrderedLists}

Now you need to induct whether we can induct new rule/rules from the given results, and your response should follow the response format. Remember that the new rule/rules should be in the new incorporated color preference rather than the original given color preference.

Figure 10: Prompt Template for Ordering in Rules Validation and Rules Incorporation