Challenging Large Language Models with New Tasks: A Study on their Adaptability and Robustness

Chenxi Li[♥*], **Yuanhe Tian**[♥*], **Zhaxi Zerong**[♥], **Yan Song**[♠†], **Fei Xia**[♥†]
[♥]University of Washington [♠]University of Science and Technology of China
[♥]{cl91, yhtian, tashi0, fxia}@uw.edu [♠]clksong@gmail.com

Abstract

Recent progress in large language models (LLMs) has marked a notable milestone in the field of artificial intelligence. The conventional evaluation of LLMs primarily relies on existing tasks and benchmarks, raising concerns about test set contamination and the genuine comprehension abilities of LLMs. To address these concerns, we propose to evaluate LLMs by designing new tasks, automatically generating evaluation datasets for the tasks, and conducting detailed error analyses to scrutinize LLMs' adaptability to new tasks, their sensitivity to prompt variations, and their error tendencies. We investigate the capacity of LLMs to adapt to new but simple tasks, especially when they diverge from the models' pre-existing knowledge. Our methodology emphasizes the creation of straightforward tasks, facilitating a precise error analysis to uncover the underlying causes of LLM failures. This strategic approach also aims to uncover effective strategies for enhancing LLM performance based on the detailed error analysis of system output.[1]

1 Introduction

LLMs have produced impressive results on many NLP tasks (Zhang et al., 2019a,b; Devlin et al., 2019; Tian et al., 2020; Touvron et al., 2023), and existing studies for evaluating LLMs mainly leverage benchmark datasets for existing tasks, such as summarization (Tian et al., 2024), question answering (Berant et al., 2013; Joshi et al., 2017; Kwiatkowski et al., 2019; Reddy et al., 2019), reasoning (Clark et al., 2018), and math word problems (Hendrycks et al., 2021; Cobbe et al., 2021; Shi et al., 2022). These evaluation approaches face the test set contamination issue (Oren et al., 2023). It is also difficult to determine to what degree the high performance of LLMs relies on the memorization of prior knowledge acquired during pretraining or on their ability to successfully follow the instructions in the prompts.

Another challenge with prior research lies in the increasing complexity of benchmark tasks. Paired with the frequent absence of thorough error analysis, the resulting performance metrics often amount to mere numbers. They lack the substantive insights necessary for understanding why models struggle with certain tasks, which is essential for improving LLMs in the future.

In this study, we focus on the following research questions: (Q1) How effectively can LLMs adapt to new tasks, particularly when instructions in the prompts conflict with the model's prior knowledge? (Q2) How robust are LLMs to minor variations of prompts? (Q3) What are the common error types made by LLMs in our designed tasks? (Q4) Drawing from the insights gained in the preceding questions, what strategies can be employed to enhance LLMs' performance on specific tasks?

To answer these questions, we propose a new paradigm of Evaluating LLMs with Automatically Generated Evaluation datasets for New Tasks (ELAGENT) - we create new tasks and new datasets instead of exploiting current benchmarks - to reduce the risk of data contamination. More importantly, our tasks are designed to be novel but simple to facilitate identifying the source of LLMs' deficiencies during error analysis. In addition, based on detailed error analysis, we improve system performance via techniques such as chain-of-thought (CoT) (Wei et al., 2022).

2 Task Design

To answer the research questions, we design six new tasks, automatically create datasets for them, design prompts, and conduct detailed error analyses on the LLM output.

* Equal contribution.
† Corresponding author.
[1]The code and processed datasets are available at https://github.com/CLINEEK/ELAGENT.

Findings of the Association for Computational Linguistics ACL 2024, pages 8140–8162

Task 1 We redefine the priorities of elementary arithmetic operators as follows: '+' > '*' = '-' Based on the new priorities, what is the value of 8 - 4 * 4 + 9? Your can give the result as a decimal, and please put the final numeric result in brackets [], such as [1.0]
Task 2 For the expression 8 - 4 * 4 + 9. the precedence of the operators is redefined. If 8 - 4 * 4 + 9 = 52, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B]. (A) '+' = '*' = '-' (B) '-' = '*' > '-' (C) '+' > '*' = '-' (D) '-' = '*' = '+'.

Table 1: Example prompts for Task 1-2.

2.1 Task 1 & 2: Arithmetic Expression Evaluation with Changed Precedence

Previous studies (Brown et al., 2020; Anil et al., 2023) have tested LLMs' performance in arithmetic expression evaluation, revealing that chain-of-thought (CoT) (Wei et al., 2022) enhances model accuracy. To answer Q1, we change the original task by redefining the precedence of elementary arithmetic operations to assess LLMs' ability with changed math rules. In Task 1, the LLM is given the new precedence and an expression and asked to produce the correct value; In Task 2, the model is given an expression and its corrected value and asked to choose the correct precedence of operators. The prompts for the tasks are shown in Table 1. Subsequently, we analyze the system output to investigate the effect of variations in prompts on model performance, addressing Q2-Q4.

We measure the difference between the new and the standard precedence by the number of 'moves' needed to change from the latter to the former. The precedence is represented in a five-slot array. Fig 1(a) shows the standard, where '*' and '/' are in slot 2, and '+' and '-' are in slot 4. Here, smaller numbers indicate higher priorities. To create a new precedence, some operators are moved from their original slots to new ones. For instance, in Fig 1(b), '+' is moved from slot 4 to 1; in Fig 1(c), '+' is from slot 4 to 1, and '*' from slot 2 to 3. We call these three cases **no move**, **one move**, and **two moves**, respectively. In Section 3.1, we report system performance with respect to each case.

2.2 Task 3 & 4: Machine Translation with an Artificial Language

Prior research on machine translation (MT) primarily focus on translation between natural languages such as English and French (Bawden and Yvon, 2023; Tamura et al., 2023; Zhang et al., 2023). To test LLMs' abilities in understanding prompts, we define a non-existing language called Language A, which is similar to English except for the syntactic or morphological properties in Table 2. In Task 3, LLMs are given an English sentence, the translation of the words, a subset of properties in Table 2 and asked to translate the English sentence into Language A. Task 4's prompts include two English sentences and their translations in Language A as well as the word-level translation pairs. LLMs are asked to infer Language A's syntactic properties (e.g., whether Language A is SOV or SVO). The prompts are shown in Table 3.

The tasks assess LLMs' prior knowledge about linguistics (e.g., the meaning of SVO) and their adaptability to new linguistic challenges such as translating English into an artificial language, tackling research question Q1. We explore LLMs' robustness with respect to prompt variations (Q2), identify common error types (Q3), and improve LLMs' performance with CoT (Q4).

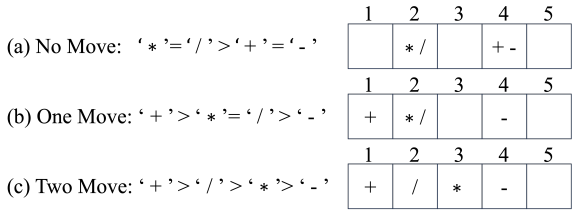

Figure 1: Redefine priorities of operators in Task 1-2 by moving operators from their original slots to new ones.

ID	Rule
S1 (SOV)	In Language A, the canonical word order of sentences is Subject-Object-Verb.
S2 (PV)	In Language A, modifiers of verbs come before the verbs. If a verb is modified by both a preposition phrase (PP) and an adverb, the correct word order is PP first, then the adverb, and then the verb.
S3 (PN)	In Language A, modifiers of nouns appear before nouns. When a noun has multiple modifiers, the word order is prepositional phrases first, then determiners, then adjectives, and finally the noun.
M1 (Plu)	In Language A, a singular noun and a plural noun start with prefix 'S_' and 'P_', respectively, followed by its lemma.
M2 (PT)	In Language A, verbs in past tense and present tense are represented by the lemma, followed by suffix '_-' and '_+', respectively.
M3 (AR)	In Language A, each adjective appears twice consecutively in a sentence.

Table 2: The set of linguistic properties that a prompt can select when defining Language A for Task 3.

Task 3
The following are the linguistic rules of Language A: <Rules>
For anything not explicitly specified, Language A follows the same word order as English.
Here are the word pairs between English and the language A: <Word Pairs>.
Translate the following simple sentence into Language A: "<English Sentence>".
Please put the final translated result in brackets [] at the end of your response.

Task 4
<Sentence X1> in English is translated to <Sentence Y1> in Language A <Sentence X2> in English is translated to <Sentence Y2> in Language A
Here are the word pairs between English and Language A: <Word Pairs>.
What is the canonical word order in Language A: (a) SVO only, (b) VSO only, (c) SVO or SOV, (d) SOV only

Table 3: Example prompts for Task 3 and Task 4. Full prompt for Task 3 can be found in Appendix G.

Predicate Types	Example of Definition
None-transitive & None-symmetrical	X fears Y
Transitive & None-symmetrical	X runs faster than Y
None-transitive & Symmetrical	X and Y are partially related
Transitive & Symmetrical	X and Y have the same speed

Table 4: Four types of predicates and examples of definition given in Task 6 prompts.

2.3 Task 5 & 6: Basic Deduction with a Twist

In Task 5, we evaluate LLMs' reasoning ability. We start with the bAbI dataset (Weston et al., 2015), which has been used by several prior studies (Xu et al., 2023; Bang et al., 2023). We adopt the basic deduction test set from bAbI; an example is shown in Table 5. To answer Q1, we modify the dataset in two ways: we replace nouns in the original statements with random strings to test LLMs' ability to handle unfamiliar entities; to answer Q2, we negate the verb or add *some* to the subject or the object of some statements to check whether LLMs interpret negation and quantifiers properly during reasoning.

In Task 6, we assess LLMs' capability to discern semantic nuances of predicates. Some predicates represent transitive relations (e.g., if X runs faster than Y and Y runs faster than Z, then X runs faster than Z); Some represent symmetric relations (e.g., "X equals Y" implies "Y equals X"). Thus, predicates can be divided into four types according to these two properties, as shown in Table 4. We want to test whether LLMs can distinguish different types of predicates during reasoning. To prevent LLMs from utilizing their prior knowledge about the actual predicates in English, we define new predicates (written as random strings) in the prompts and ask LLMs to reason about the predicates. The example prompts of Task 5-6 are in Table 5.

Task 5
(1) Wolves are afraid of mice. (2) Sheep are afraid of mice. (3) Winona is a sheep. (4) Mice are afraid of cats. (5) Cats are afraid of wolves. (6) Jessica is a mouse. (7) Emily is a cat. (8) Gertrude is a wolf.
Based on the above sentences, what is Emily afraid of?
(a) cat (b) mouse (c) wolf (d) sheep (e) not enough information is provided to answer the question

Task 6
'X BBTuL Y' means 'X runs faster than Y'. Suppose (1) X2 BBTuL X3, (2) X4 BBTuL X1, and (3) X1 BBTuL X2. What does X1 BBTuL?
(a) Only X2 (b) Only X3 (c) Only X4 (d) X2 and X3 (e) X2 and X4 (f) X3 and X4 (g) X2, X3, and X4

Table 5: Example prompt for Task 5 and 6. Task 5 uses a revised version of the bAbI dataset (Weston et al., 2015); Task 6 uses our own newly created datasets.

2.4 Dataset creation

We begin Task 5 with the basic deduction test set sourced from the bAbI dataset. Subsequent modifications to Task 5 and the creation of datasets for the other five tasks are all automated, including the generation of prompts, gold standards, and evaluations. The detailed process of dataset creation is included in Appendix C.

3 Experiment Setting

We run the experiments on GPT4-Turbo[2] (OpenAI, 2023), LLaMA-2-chat-70B (LLaMA2) (Touvron et al., 2023), LLaMA-3-70B-Instruct (LLaMA3) (AI@Meta, 2024), and Flan-T5-XXL (T5) (Raffel et al., 2020). For comparison, we also randomly select some instances from each task's dataset and ask human annotators to answer the questions.

3.1 GPT4 Setting

In the main section of the paper, we have opted to exclusively report the experiments conducted with GPT4, given its superior performance compared to other models (see Appendix A). Similarly, our error analysis also centers on the output generated by GPT4.

[2]We use their API at https://platform.openai.com

3.1.1 Non-determinism of GPT4

GPT4 model is known to be non-deterministic (Ouyang et al., 2023) even when its hyperparameter top_p is set to 0, which requires the model to always select the token with the highest probability when generating the next token. We have run the same experiments multiple times and confirmed that the model we use, GPT-4-preview-1106, is indeed non-deterministic. On the other hand, we found that the system performance does not vary much with multiple runs. We report the mean and the standard deviation of system performance on each task in Appendix A.

3.1.2 Sampling Strategies

To explore the performance of GPT4 with varying top_p values, we run Task 1 experiments with four top_p values on two-move expressions under the 1-shot-CoT setting. The results are in Table 6, which shows that the system performance does not vary greatly with respect to different top_p values. Thus, we have opted to utilize top_p=0 for all the main experiments reported in this paper.

	0	**0.1**	**0.2**	**0.4**
1d	90.2±0.5	91.5±1.2	91.5±1.0	92.0±1.5
2d	94.7±1.2	93.2±1.3	94.8±0.5	94.5±1.5
3d	78.0±0.7	78.0±0.4	77.0±1.5	77.8±0.2
4d	48.3±1.0	48.7±1.2	49.0±0.7	47.3±2.0
5d	45.7±0.6	44.8±1.0	44.8±0.5	45.3±0.8

Table 6: Accuracy of GPT4 on two-move expressions in Task 1 under the 1-shot-CoT setting. The row label shows the number of digits in the operands; the column labels are top_p values; temperature is fixed at 0.4. Each cell reports the mean and standard deviation of accuracy with three runs.

3.2 Other models

Beside GPT4, we also evaluate these tasks on LLaMA2, LLaMA3, and T5. Notably, the performance of T5 demonstrates significant deficiencies. Its output is often incomplete or erroneous, so we choose not to report T5 results. Both LLaMA2 and LLaMA3 exhibit lower performance compared to GPT4 in almost all tasks. A full performance comparison of the three models is in Appendix A.

3.3 Human Evaluation

When we design the tasks, we want to keep them simple and easy, at least to humans. To determine whether that is the case, we ask human annotators to answer a subset of questions in our datasets. These responses are included for comparison with model performances.

Among the six tasks, Task 3 is the most time-consuming one and its exact match score is very strict. Thus, we randomly select 10 instances for human evaluation. For all other tasks, we randomly selected 50 instances from each dataset. All the prompts given to the human annotators are without CoT instructions. The results are in Appendix A, alongside the performance of GPT4, LLaMA2 and LLaMA3.

4 GPT4 Results

In all the tables in this section that report system performance, each cell reports accuracy on 200 test instances except for Task 5, which uses 250 test instances modified from the original bAbI dataset.

For Task 1-3, we experiment with chain of thought (CoT) with 0-shot and 1-shot. In this section, we include only the results with 0-shot without CoT (denoted as **0-s**) and 1-shot with CoT (denoted as **1-s-c**). More results including the ones under the settings of 0-shot-with-CoT and 1-shot-without-CoT are in Appendix B. The full prompts and some system output for Task 1-3 are in Appendix C-E.

4.1 Task 1: Arithmetic Expression Calculation

For Task 1, we run the experiment under the zero-move, one-move, and two-move settings (as defined in Sect 2.1); each operand has one to five digits. The result is in Table 7, which shows that the system performance decreases as the number of digits in the operands increases, consistent with previous findings (Brown et al., 2020; Touvron et al., 2023). The system performance also deteriorates significantly when the precedence is different from the standard one (e.g., the accuracy of two digits calculation for two moves dropped drastically from 95.0% to 39.5%) under the 0-shot setting.

After a thorough error analysis (see Section 5.3), we identified common error types and designed CoT instructions accordingly, which greatly improved model performance (see Row 2/4/6 vs. Row 1/3/5 in Table 7). The full CoT prompt and a system output example are in Appendix D-E.

ID	Move	AP	1D	2D	3D	4D	5D
1	zero	0-s	99.0	95.0	67.5	31.5	20.0
2	move	1-s-c	100	98.5	69.0	28.0	20.5
3	one	0-s	75.0	58.5	40.5	17.5	14.5
4	move	1-s-c	94.5	97.5	73.0	35.5	32.5
5	two	0-s	54.0	39.5	32.5	15.5	13.5
6	moves	1-s-c	92.0	94.0	79.5	48.5	46.0

Table 7: Accuracy of GPT4 on Task 1. Each test instance is an arithmetic expression with three distinct operators and four operands. 'AP' denotes the approach: '0-s' means '0-shot'. '1-s-c' means 1-shot-CoT. Col 1D-5D refers to number of numbers in operands, ranging from 1 to 5. The same sets of test instances are used for the cells in the same column, but the precedence of the operators are different between no move, one move, and two moves. See Table 27 in Appendix B for the full results on Task 1.

	1D	2D	3D
0-s	25.0/52.0	21.5/61.0	19.0/62.0
1-s-c	41.5/53.0	43.5/53.0	30.5/64.5

Table 8: Performance of GPT4 on Task 2 under two-move configuration. Each cell has the format x/y, where x is the system accuracy and y is the percentage of *invalid* answers (i.e., GPT4 does not pick any choice).

4.2 Task 2: Choosing Precedence of Operators

In Task 1, we observed GPT4's subpar performance with redefined '2-move' precedence under 0-shot setting. We report GPT4's performance with such precedence on Task 2 in Table 8. The correct answer is uniformly distributed among the four choices, so the accuracy of a random guess would be 25%. Notably, the '0-shot' accuracy is worse than random guess. This is primarily caused by GPT4's low performance on Task 1 and its tendency of not picking any given choice and hence producing an invalid answer.

Like in Task 1, we experimented with 1-shot-CoT for Task 2, which leads to better performance. Nevertheless, GPT4's performance is still very low as it produces invalid answers more than half of the time, indicating that GPT4 cannot handle redefined precedence well in this task. The CoT prompts and GPT4 output examples are in Appendix F.

4.3 Task 3: Machine Translation

In Task 3, LLMs are asked to translate some English sentences into Language A. The results are in Table 9. S and M are the number of syntactic and morphological rules used when describing Language A in the prompts.

	EM	BLEU	METEOR			F1
			Score	C	UA	
(a) S=0 M=0						
0-s	100	100	100	1.00	0.0	-
(b) S=3 M=0						
0-s	66.0	97.0	94.1	2.08	0.6	-
1-s-c	93.0	99.8	99.8	1.27	0.01	-
(c) S=0 M=3						
0-s	30.0	95.5	89.5	2.56	1.59	87.9
1-s-c	49.0	98.6	95.6	1.51	0.64	93.1
(d) S=3 M=3						
0-s	1.0	88.8	68.0	5.58	4.45	59.3
1-s-c	33.5	97.3	93.4	2.10	1.11	88.9

Table 9: The performance of GPT4 on Task 3 with two types of prompts. The full results are in Table 29.

For evaluation, we use four metrics: Exact Match (EM), BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and Morph F1-score. EM is the strictest, requiring the entire translation sentence produced by the system to be identical to the gold standard; BLEU and METEOR are more lenient (for instance, failing to "apply" the M3 rule will affect only the adjectives and the ngrams that contain the adjectives). Morph F1 evaluates LLMs' ability to apply morphological rules by calculating precision and recall of the correct spellings of the words which the morphological rules should apply to. For METEOR, in addition to Score, we include aligned chunk numbers (C) and unaligned word count (UA) numbers, for which smaller numbers are better. For the other metrics, larger numbers are better.

Table 9 shows that GPT4 produces perfect translation when Language A has the same word order as English and no morphological rules is applied. However, the introduction of linguistic rules led to a decline in performance, notably more severe for strict metrics like EM, which plummeted from 100% in the no-rule-applied scenario (S=0 M=0) to 1% when three syntactic and three morphological rules are used to define language A. When comparing the impact of syntactic (S=3 M=0) vs. morphological (S=0 M=3) rules, the system's performance dipped more sharply with morphological complexities, highlighting handling morphological rules is more challenging for GPT4. Finally, 1-shot-CoT provides a big boost like in Task 1-2; the full CoT prompt is in Appendix G.

	Original Exp.			Switch Exp.		
	SOV	**PV**	**PN**	**SOV**	**PV**	**PN**
1-order	83.0	69.5	97.5	87.0	78.5	97.5
1-order-noise	95.0	66.0	98.0	91.0	64.5	96.0
2-order	2.5	17.5	14.0	7.0	77.5	88.5
2-order-noise	3.0	17.0	14.5	14.0	62.0	75.5

Table 10: GPT4's accuracy on Task 4. See Section 2.2 and Section 4.4 for the meaning of the row and column headers.

4.4 Task 4: Inferring Word Order

In Task 4, we assess LLMs' capability to identify Language A's syntactic properties when given two (English, Language A) sentence pairs. Suppose the test question is what is the canonical word order in Language A. There are four scenarios based on the word order in the two Language-A sentences:

1-order: Both language-A sentences have SOV order, and other word orders (e.g., noun and its modifier) are the same as in English.

1-order-noise: Both language-A sentences have SOV order, but other word orders are different from English (e.g., in Language A, nouns follow its PP modifier).

2-order: One language-A sentence is in SOV order, and the other is in SVO. Other word orders are the same as in English.

2-order-noise: One language-A sentence is in SOV order, and the other is in SVO; other word orders are different from English (e.g., nouns follow its PP modifier).

For the latter two scenarios, the order of the two sentence pairs matters. In our original experiments, the first language-A sentence is in SOV, and the second SVO. If we switch the order of the sentence pairs, GPT4's performance increases. The improvement is more notable if the test question is about PV or PN rules, as shown in Table 10. The table shows that GPT4 achieves relatively high accuracy in the 1-order setting and can handle noise in the 1-order-noise setting. Under the 2-order scenario, rather than choose the option that says both orders are possible, GPT4 often chooses the option of one order only, leading to much worse performance. We will discuss this in more details in Section 5.

4.5 Task 5: Basic Deduction

In Task 5, we run experiments on the original bAbI dataset and its modified versions. The accuracy for task 5 is reported in Table 11.

GPT4 achieved an accuracy of 90.8% on the original dataset. When we test GPT4 on 250 new test examples whose length of inference chains is three (increased from two in the original dataset), its performance decreased to 85.6%. Replacing nouns with random names or adding quantifiers actually increase the accuracy a little bit, which could be surprising as those changes should conceptually make the task harder. Upon detailed analysis of errors made on the original test, we have found an explanation, which is discussed in Section 5.

	Base	**Ran**	**Neg**	**Quan**	**Ext**
Acc	90.80	95.20	98.80	91.20	85.60

Table 11: Accuracy of GPT4 on Task 5. 'Base': original dataset from bAbI; 'Ran': replace nouns with random strings; 'Neg': negation some predicates; 'Quan': add 'some' to the subject of the sentence; 'ext': extend the reasoning chain.

	NT & NS	**NT & S**	**T & NS**	**T & S**
Ran Pred	99.00	87.00	83.00	52.00
Ran Ent	100.00	100.00	96.00	56.50

Table 12: Accuracy of GPT4 on Task 6. 'T': transitive predicate; 'S': symmetrical predicate. 'NT': predicate that is not transitive. 'NS': predicate that is not symmetrical. 'Ran Pred' is when the predicate in the statements (e.g., 'runs faster than') is replaced by a random string; 'Ran Ent' is when the entities in the statements (e.g., "Emily") are replaced by some random strings.

4.6 Task 6: Deduction with Different Types of Predicates

Table 12 shows GPT4's performance on Task 6 with four types of predicates. In the first row, we replace the spelling of the predicate (e.g., 'runs faster than') with a random string and add the definition of the random string to the prompt (see Table 5). In the second row, we replace the spellings of the entities (e.g., 'Emily') with random strings. Clearly, the former posts more challenges to GPT-4 than the latter.

The predicate type also affects system performance. For predicates that are not transitive nor symmetric ('NT & NS'), GPT4's performance is exceptional, reaching 99% accuracy. In contrast, GPT4 performs much worse when predicates are both transitive and symmetric ('T & S'). The performances for 'NT & S' and 'T & NS' are in between. This experiment indicates GPT4 is relatively weak at considering the properties of predicates during reasoning, especially when the predicates are replaced by random strings.

5 Discussion

We have conducted extensive error analysis across tasks. In this section, we synthesize the findings from the error analysis to answer our research questions.

5.1 Q1: How effectively can LLMs adapt to new tasks?

Through detailed error analysis of Task 1, 4, and 5, we have discovered that GPT4 is indeed prone to errors when the prompt contains information in conflict with prior knowledge. In Task 1, for two-move expression with two-digit operands, without CoT, GPT4 made 73 mistakes out of 200 instances in which it still uses the standard precedence despite it is given a redefined precedence.

Likewise, in Task 4, one or both of the Language-A sentences in the prompt have different word orders from English. Even when GPT4 correctly reiterated the Language-A sentences from the prompt, it may still claim Language A has the same word order as in English. Table 18 shows that there are 35 cases of this error type out of 200 for the rule 'prepositional phrase before noun' in Language A.

	Factual	Counterfactual
Correct	11	8
Incorrect	0	9

Table 13: Raw count of correct and incorrect predictions made by GPT4 on the original dataset for Task 5 when one evidence statement is factual vs. counterfactual.

In Task 5, we find that the errors on the original dataset are correlated with the counterfactuality condition in the input statements. For instance, the factual statement 'Mice are afraid of Cats' is used as the evidence in the reasoning chain 11 times, and GPT4 picks the correct answer every time. In contrast, the counterfactual statement 'Cats are afraid of mice' is used 17 times, and GPT4 picks the correct answers only 8 times. The accuracy count is shown in Table 13. Fisher's exact test returns a p-value of 0.004, implying that counterfactual statements, namely 'Cats are afraid mice', often lead to errors when used as reasoning evidence.

Finally, a comparison between Table 11 and 12 shows the LLMs are good at understanding negation and quantifier, but much worse at discerning the semantic nuances introduced by transitivity and symmetry, which are concepts that might not have been learned well by the LLMs.

5.2 Q2: How robust are LLMs to minor variations of prompts?

Based on our experiments, LLMs are robust in handling some types of noise in the prompt. As evidenced by Table 10 from Task 4, the presence of noise in prompts has a minimal impact on LLMs' abilities.

	SOV	SVO	VSO	SVO&SOV
SOV	0	0	0	0
SVO	0	0	0	0
VSO	0	0	0	0
SVO&SOV	155	39	0	6

Table 14: Confusion Matrix for Task 4's 2-Order **Original Experiment** asking about Language A's canonical word order. In the experiment, the language-A sentences in the two sentence pairs are in SOV and SVO, respectively. In the table, row labels represent the correct answers, while column labels indicate GPT4's predictions.

	SOV	SVO	VSO	SVO&SOV
SOV	0	0	0	0
SVO	0	0	0	0
VSO	0	0	0	0
SVO&SOV	37	135	0	28

Table 15: Confusion Matrix for Task 4's 2-order **Switch Experiment** asking about Language A's canonical word order. Row labels represent the correct answers, while column labels indicate GPT4's predictions. All the system setup and the test instances for this experiment are identical to the ones in the **Original Experiment** in Table 14 except that the order of the two sentence pairs in each test instance is switched.

However, they are not robust but rather sensitive to the positioning of the key information in the prompt. For instance, models' performance increased dramatically when the two sentence pairs are switched in Task 4. Table 14 shows the confusion matrix for the 2-order setting when the test question asks about canonical word order and the first language-A sentence is in SOV order. Table 15 is the confusion matrix when the order of the two sentence pairs are switched. The contrast between the two tables show how sensitive GPT4 is to the position of certain input.

This observation was validated through a chi-square analysis detailed in Table 16, which mea-

	SOV	SVO	SVO&SOV	SubTotal
SOV 1st	155	39	6	200
SVO 1st	37	135	28	200
Total	192	174	34	400

Table 16: Chi-square test of SOV Rule Recognition Before and After Sentence Pair Reordering in Tables 14 and 15. 'SOV 1st' indicates scenarios where the first sentence pair follows the SOV structure, and conversely for the switched condition. p-value is lesser than 0.00001.

sured the impact of sentence pair sequencing on the LLM's predictive accuracy. The results confirmed a pronounced correlation, emphasizing the significant role that the sequence of presentation plays in the LLM's capacity to correctly identify and apply syntactic rules.

Similarly, the error analysis of Task 5 reveals that errors are correlated with the position of the reasoning evidence in the given eight statements. Among the eight statements in the prompt, if the first statement is part of the reasoning chain, GPT4 is more likely to make mistakes. To be more specific, out of 23 errors, 13 use first statement in the reasoning chain. In the total 250 instances, 93 use the first statement, and the rest 157 do not. The error rate with regard to the smallest position of the evidence sentence is reported in Table 17. Both analyses show that the different placements of key information in the prompt will affect model performance.

	Incorrect	Total	Error Rate
1st stmt	13	93	14.0%
Rest stmts	10	157	6.4%

Table 17: The count of correct and incorrect instances out of 250 with regard to the position of the evidence in the prompt.

5.3 Q3: What are the common error types made by LLMs in our designed tasks?

We have conducted comprehensive error analysis on Task 1 and Task 4.

In Task 1, Errors for two-move expressions with two digits without CoT can be classified into following types[3]: (1) false precedence (FP): the model succeeds in translating the newly defined precedence into words but still uses the standard in the subsequent calculations; (2) false interpretation (FI): the model fail to translate the newly defined precedence into words or fail to understand the expression correctly; (3) wrong operands (WO): the model fails to evaluate an operator with operands next to it; (4) missing step (MS): the model fails to execute one step of calculation due to ignoring one operator; (5) extra step (ES): the model adds one more step of calculation. The raw count of these error types are detailed in 2[4].

[3]We omit the error types, such as miscalculation, that have count of one.

	MR1	MR2	FO1	FO2
original	38	92	35	7
switch	17	2	1	3

Table 18: Error Type Distribution for "PP Before Noun" Rule in Task 4: Analyzing 200 Instances. "MR" denotes misremembered translations, with MR1 and MR2 indicating errors in the first and second sentence pairs, respectively. "FO" signifies incorrect observations, distinguishing between the first (FO1) and second (FO2) translated sentences.

In Task 4, our error analysis, as detailed in Table 18, identifies two primary error types made by LLMs. The first type involves misremembering the translated sentence from the prompts, labeled as MR errors. Here, LLMs inaccurately replicate the sentence in Language A, either altering the word order of the first translated sentence (MR1) or modifying the second translated sentence (MR2). An example of this error is when the original prompt sentence "the dog with a tail" is incorrectly echoed by the LLM as "with a tail the dog."

The second type of error, referred to as FO errors, occurs when LLMs correctly copy the translated sentence but then make incorrect observations about it. These errors are divided into two categories: faulty observations about the first translated sentence (FO1) or the second translated sentence (FO2). For example, despite accurately repeating "the dog with a tail" from the prompts, the LLM might erroneously analyze this structure as "prepositional phrase before noun."

5.4 Q4: What strategies can be employed to enhance LLM performance on our tasks?

Drawing on the previously mentioned error types and the acknowledged effectiveness of CoT, a promising strategy to boost the performance is to

[4]We count multiple errors in one instance separately.

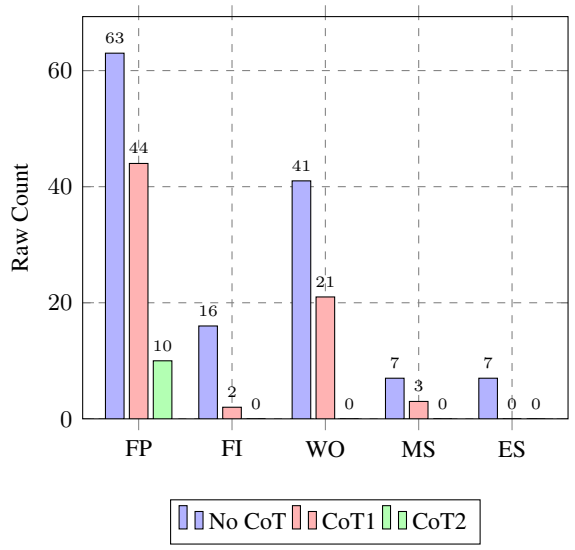

Figure 2: Raw count with regard to error types across No CoT, CoT1, and CoT2 conditions for Task 1. The explanation of five types is in Section 5.3.

craft prompts tailored to guiding models in avoiding those common errors. CoT prompts is a known approach to boost model performance, but we propose that CoT could achieve the most desirable effect when designed in accordance with the results of detailed error analysis. After concluding the error types for Task 1, we design two versions of CoT prompt. The result is shown in Fig 2. The first version (CoT1) prompt that instructs the model to interpret the new precedence and evaluate in steps, but the positioning of operators in the question expression is different from that in the demonstration example. CoT1 greatly reduced errors of FI, MS, and ES, but not FP. We then design CoT2 to ensure that this positioning is also aligned. The improved CoT drastically reduces errors caused by FP, and eliminates other error types.

6 Related Work

Numerous studies have been conducted to assess LLMs. Many have extensively examined a diverse range of LLM abilities on existing benchmarks (Laskar et al., 2023; Bang et al., 2023; Qin et al., 2023), but the results may be influenced by the data contamination problem (Brown et al., 2020).

Several studies have integrated strategies to mitigate such risks in dataset development. For instance, Wu et al. (2023) devised counterfactual tasks that are less likely to have been encountered by LLMs, while Saparov and He (2022) constructed datasets featuring false and fictional ontologies, which are potentially unfamiliar to LLMs. Our work differs from theirs in that we designed six new tasks and conducted a comprehensive error analysis to identify common error types made by LLMs on these tasks.

In terms of the robustness of LLMs, MUCH attention IS focused on evaluating their resistance to adversarial attacks (Zhao et al., 2021; Hou et al., 2024). A related study explores the effect of reordering answer choices in multiple-choice questions on model performance (Pezeshkpour and Hruschka, 2023). We test the robustness of LLMs in several ways, such as randomizing the spellings of words (in Task 3-6) and altering the placement of key information (in Task 4-5).

7 Conclusion

In this study, we propose a new paradigm for evaluating LLMs, which includes three crucial components: designing simple tasks, automatically generating the evaluation dataset, improving system performance based on thorough error analysis. Our experiments show LLMs such as GPT4 struggle with simple tasks when instructions conflict with prior knowledge. Our research also highlights LLMs' robustness to some types of noise but vulnerability to other variations, such as the position of key information in the prompts. Additionally, we have shown that extensive error analysis can guide the design of CoT by aiming at eliminating common errors and improving model performance.

The six tasks used in this study cover a small portion of the spectrum in assessing LLMs. Therefore, we intend to develop additional tasks to explore other facets of LLM capabilities. Also, while our study focuses on analyzing the output of GPT4, we aim to extend our analysis to include other models. By conducting detailed error analyses on the outputs of these models, we hope to identify common patterns in errors across different LLMs. This approach could potentially offer insights into the underlying architectures of LLMs and contribute to their further development.

Acknowledgement

We would like to express our sincere appreciation to Yian Wang, Yutong Li, and Chenxin Liu for providing human annotation for the six tasks. We would also like to thank ARR reviewers for their valuable comments on this paper. Lastly, we would like to thank Shane Steinert-Threlkeld for his constructive feedback on this paper.

References

AI@Meta. 2024. LLaMA 3 Model Card. Https://github.com/meta-llama/llama3.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. PaLM 2 Technical Report. *arXiv preprint arXiv:2305.10403*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Rachel Bawden and François Yvon. 2023. Investigating the translation performance of a large multilingual language model: the case of bloom. *arXiv preprint arXiv:2303.01911*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-shot Learners. *Advances in neural information processing systems*, 33:1877–1901.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think You Have Solved Question Answering? Try ARC, The AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving with the Math Dataset. *arXiv preprint arXiv:2103.03874*.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, pages 364–381. Springer.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Xiangji Huang. 2023. A Systematic Study and Comprehensive Evaluation of ChatGPT on Benchmark Datasets.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yonatan Oren, Nicole Meister, Niladri Chatterji, Faisal Ladhak, and Tatsunori B Hashimoto. 2023. Proving Test Set Contamination in Black Box Language Models. *arXiv preprint arXiv:2310.17623*.

Shuyin Ouyang, Jie M Zhang, Mark Harman, and Meng Wang. 2023. LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation. *arXiv preprint arXiv:2308.02828*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Pouya Pezeshkpour and Estevam Hruschka. 2023. Large language models sensitivity to the order of options in multiple-choice questions. *arXiv preprint arXiv:2308.11483*.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits

of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A Conversational Question Answering Challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language Models are Multilingual Chain-of-Thought Reasoners. *arXiv preprint arXiv:2210.03057*.

Hiroto Tamura, Tosho Hirasawa, Hwichan Kim, and Mamoru Komachi. 2023. Does Masked Language Model Pre-training with Artificial Data Improve Low-resource Neural Machine Translation? In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2216–2225, Dubrovnik, Croatia.

Yuanhe Tian, Yan Song, and Fei Xia. 2020. Supertagging Combinatory Categorial Grammar with Attentive Graph Convolutional Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6037–6044.

Yuanhe Tian, Fei Xia, and Yan Song. 2024. Dialogue Summarization with Mixture of Experts based on Large Language Models. In *The 62nd Annual Meeting of the Association for Computational Linguistics*, Bangkok, Thailand.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. LLaMA 2: Open Foundation and Fine-tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. *arXiv: Artificial Intelligence*.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2023. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2023. Are large language models really good logical reasoners? a comprehensive evaluation from deductive, inductive and abductive views. *arXiv preprint arXiv:2306.09841*.

Hongming Zhang, Yan Song, and Yangqiu Song. 2019a. Incorporating Context and External Knowledge for Pronoun Coreference Resolution. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 872–881.

Hongming Zhang, Yan Song, Yangqiu Song, and Dong Yu. 2019b. Knowledge-aware Pronoun Coreference Resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 867–876, Florence, Italy.

Xuan Zhang, Navid Rajabi, Kevin Duh, and Philipp Koehn. 2023. Machine translation with large language models: Prompting, few-shot learning, and fine-tuning with QLoRA. In *Proceedings of the Eighth Conference on Machine Translation*, pages 468–481.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

A Full Comparison of GPT4, LLaMA2, LLaMA3, and Human Performance

In this appendix, we include the full comparison of the performance between GPT4, LLaMA2, LLaMA3, and human evaluation over the six tasks. Due to the non-deterministic nature of GPT4, we also run the experiment three times for each cell with temperature set to 0.4 and top_p to 0 and report the mean and standard deviation. Again, the results of T5 are not reported because the system output is either incomplete or erroneous. The human evaluations reported in the following tables are all on prompts without CoT instructions. The comparison of Task 1 is detailed in Table 19, of Task2 is detailed in Table 20, of Task 3 is detailed in Table 21, of Task 4 is detailed in Table 22, of Task 5 is detailed in Table 24, and of Task 6 is detailed in Table 25.

B Additional results of GPT4 on the six tasks

In this appendix, we report any additional experiment results of GPT4 that we are unable to include in the main section due to space limit. The full results of GPT4 performance on Task 1 is detailed in 27. The full results on Task 3 is detailed in Table 29.

C Detailed process of dataset creation

Following the task design ideas and the template finalized in the prompt adjustment, we then create dataset for each task. In each subset of data for Task 1 - Task 4 and Task 6, we create 200 test instances. Each subset for Task 5 contains 250 instances, which are adopted and modified from the bAbI dataset. The creation is detailed in the following sections in this chapter.

C.1 Task 1

For Task 1, a test instance is an *(exp-priorities, val)* pair, where *exp* is an arithmetic expression, *priorities* specify the priorities of the operators in the expression, and *val* is the correct value of the expression with the specified priorities. During the evaluation, *exp* and *priorities* are inserted into the prompt template, and the system output is compared with *val* to calculate the accuracy. We generate an expression in three steps:

- (1) Pick the number of operators, n, in the expression; in this study we will focus on the cases when n is 2 or 3.
- (2) Randomly sample (with replacement) $n+1$ operands from all the positive integers with m digits, m ranges from 1 to 5.
- (3) Randomly choose n distinct operators from the four basic operators ('+', '-', '*', and '/') to connect the $n+1$ operands. We divide the expressions into 2*5=10 subsets based on the values of n and m.

To generate priorities of the operators in the expression, when n is 2, the two operators in an expression (e.g., *2+3*4*) follow either the standard priorities (e.g., '*' > '+') or the opposite one (e.g., '+' > '*'). They correspond to the *no-move* and *one-move* scenarios, respectively. When n is 3, there are six possible orders for evaluating the expression. For the simplicity of presenting evaluation results, we pick three of the six possible orders: one with the standard priorities (i.e, *no move*), one with *one move*, and the last one with *two moves*.

Once we have redefined the priorities, we evaluate the expression based on the priorities to get *val*. We keep an expression in our dataset only if its values differ when the priorities change.[5] For

[5]E.g., we will throw away 2+3*1 because its answer is always 5 regardless of which operator has higher priority.

	No CoT			CoT			Human
	GPT4	LLaMA2	LLaMA3	GPT4	LLaMA2	LLaMA3	
0-move	96.7 ± 0.8	18.0	9.5	98.8 ± 0.3	46.0	20.5	100
1-move	57.3 ± 1.6	1.5	2.0	96.7 ± 0.8	31.0	11.5	98.0
2-move	47.7 ± 4.3	2.0	1.5	94.5 ± 1.5	31.5	11.5	100

Table 19: Accuracy comparison between GPT, LLaMA2, LLaMA3, and human evaluation on Task 1 for two-digit expressions. Each cell reports the accuracy of *n* instances. *n* is 50 for human evaluation and 200 for models.

	No CoT			CoT			Human
	GPT4	LLaMA2	LLaMA3	GPT4	LLaMA2	LLaMA3	
one digit	25.0 ± 0.9	12.5	12.5	39.2 ± 2.0	4.5	17	98.0
two digits	23.2 ± 1.5	15.0	9.5	43.5 ± 2.2	4.0	19	100

Table 20: Accuracy comparison between GPT4, LLaMA2, LLaMA3, and human evaluation on Task 2 under two-move configuration. Each cell reports the accuracy of *n* instances. *n* is 50 for human evaluation and 200 for models. The CoT does not work well with LLaMA2 because it become confused about the input prompt when the input is very long. For example, LLaMA2 sometimes outputs 'I agree that C is the correct answer' as if it is responding to the answer shown in the demonstration example, while it should be responding to the actual question.

each (n, m) pairs, our dataset consists of 200 expressions with the redefined priorities and corresponding values.

C.2 Task 2

The dataset for Task 2 is derived from the one with Task 1 with two changes. First, for each test instance *(exp-priorities, val)* in Task 1, we randomly pick three wrong choices of the precedence of the operators in *exp* such that those choices lead to expression values different from *val*. Second, we put the correct and wrong choices in a list, randomly shuffle it, and then assign the choices as Choice A, B, C, and D in a multiple-choice question.

C.3 Task 3

The dataset for Task 3 begins with a manually created seed sentence that contain all six rules in Table 2: 'The yellow dog with a small tail chases the black cat quickly across the park'. We create a test set for Task 3 in the following steps.

- (1) We add the seed sentence to a prompt and ask GPT4 to generate 200 sentences with similar structures.

- (2) We use SpaCy[6] to parse these sentences to obtain the dependency structure. For sentences that are ambiguous or whose parse trees are incorrect (e.g., with wrong PP attachment), we make minor modifications to the sentences to remove the ambiguities and ensure that the resulting parse trees by SpaCy are correct.[7]

- (3) For each (m, n) pairs (where m and n ranges from 0 to 3), we create a dataset with the same 200 test instances; each instance consists of an English sentence, m randomly selected syntactic rules and n randomly selected morphological rules from Table 2, and automatically generated gold standard translation by applying syntactic rules to the dependency structure and morphological rules to related words (e.g., M1 to a singular or plural noun).

C.4 Task 4

Task 4's dataset is built upon Task 3. In task 4, Each test question asks LLMs to infer one word order given two sentence pairs, the English sentence and its Language-A translation. The sentence pairs are selected from different datasets in Task 3 to create the following four scenarios. Suppose the test question is 'what is the canonical word order in Language-A'. The four scenarios based on the word order in the two Language-A sentences are:

- **1-order**: Both language-A sentences have

[6] https://spacy.io/

[7] For instance, suppose the sentence is 'NP1 V NP2 PP Adv', where PP should attach to the verb, but the parser mistakenly attach it to NP2. We just switch the positions of PP and Adv to get the new order 'NP1 V NP2 Adv PP', which SpaCy can parse correctly.

	No CoT			CoT			Human
	GPT4	LLaMA2	LLaMA3	GPT4	LLaMA2	LLaMA3	
0S0M	100 ± 0.0	12.0	4.0	-	-	-	-
3S0M	65.9 ± 0.9	0.0	0.0	95.6 ± 1.9	0.0	0.0	100
0S3M	31.1 ± 4.2	0.0	0.0	48.1 ± 1.8	0.0	0.0	100
3S3M	1.0 ± 0.4	0.0	0.0	33.4 ± 0.9	0.0	0.0	90.0

Table 21: Accuracy comparison between GPT, LLaMA2, LLaMA3, and human evaluation on Task 3 for different number of sentence rules and morphological rules. Each cell reports the exact match of n instances. n is 10 for human evaluation and 200 for models. Note that for human annotators regard prompts with or without CoT the same questions.

	Original			Switch		
	SOV	**PV**	**PN**	**SOV**	**PV**	**PN**
GPT4	6.9 ± 2.8	16.5 ± 0.9	16.1 ± 1.2	14.8 ± 1.4	62.5 ± 2.7	76.0 ± 1.3
LLaMA 2	81.1	21.9	30.4	79.2	21.4	30.8
LLaMA 3	40.4	67.8	72.3	39.6	74.4	64.9
Human	100	100	98	98	100	100

Table 22: Comparison of accuracy for GPT4, LLaMA 2, LLaMA 3, and human evaluations on Task 4 involving 2-order-noise; column title adopted from Table 10.

SOV order, and other word orders (e.g., noun and its modifier) are the same as in English.

- **1-order-noise**: Both language-A sentences have SOV order, but other word orders are different from English (e.g., in Language-A, nouns follow its PP modifier).
- **2-order**: One sentence is in SOV order, and the other is in SVO. Other word orders are the same as in English.
- **2-order-noise**: One sentence is in SOV order, and the other is in SVO; other word orders are different from English (e.g., nouns follow its PP modifier). All the answer choices in Task 4 is fixed. For example, for prompt that asks about canonical order of Lanaguge A, the four answers are fixed as '(a) SVO only, (b) VSO only, (c) SVO or SOV, (d) SOV only'.

C.5 Task 5

For Task 5, we adopt the basic deduction test set from bAbI dataset and change the question format from question answering to multiple choice question to make the gold answer unique.

In the original prompt, there are eight statements of two sentence structures: (1) <Person Name> is a <Animal Name>, (2) <Animal Name 1> is afraid of <Animal Name 1>. The question is 'What is <Person Name> afraid of?' The gold answer is a type of animal. In order to arrive at the gold answer, two steps of reasoning chain are involved. For instance, for the example displayed in Table 5, to answer 'what is Emily afraid of?' The two steps - 'Emily is a cat' and 'Cats are afraid of wolves' - will obtain the gold answer of 'wolf'. However, from the small-scaled test, GPT4 produces as the final answer 'Gertrude', which should also be acceptable as 'Gertrude is a wolf' in this example. To make the gold answer unique, we modified the original prompt into multiple question format.

Additional datasets are created by modifying each test instance in the original dataset detailed in Section 2.3. For random string dataset, we replace all the person names and animal names in the prompts with random strings of length four to six, and leave everything unchanged. For negation and quantifier dataset, we add 'not' or 'some' to the evidence sentences used in the reasoning chain respectively to alter the reasoning chain. For extended dataset, we change the answer choices from animal names to person names so that the deduction requires one more step of '<Person Name> is a <Animal Name>' to arrive at the answer.

C.6 Task 6

For Task 6, we firstly create a template 'Suppose (1) X1 p X2, (2) X2 p X3, and (3) X4 p X1, what

		1 order		1-ord-noise		2-order		2-ord-noise	
		Orig. Exp.	Sw. Exp.	Orig. Exp.	Sw. Exp.	Orig. Exp.	Sw. Exp.	Orig. Exp.	Sw. Exp.
GPT4	**SOV**	83.0	87.0	95.0	91.0	2.5	7.0	3.0	14.0
	PV	69.5	78.5	66.0	64.5	17.5	77.5	17.0	62.0
	PN	97.5	97.5	98.0	96.0	14.0	88.5	14.5	75.5
LLaMA 2	**SOV**	9.9	8.4	0.0	0.5	78.0	81.2	81.1	79.2
	PV	5.0	6.5	6.3	5.5	22.3	20.9	21.9	21.4
	PN	3.7	4.8	7.2	10.3	26.3	20.5	30.4	30.8
LLaMA 3	**SOV**	22.2	21.3	9.8	14.4	27.3	29.8	40.4	39.6
	PV	9.0	15.5	15.9	13.6	61.8	67.4	67.8	74.4
	PN	13.7	14.4	12.0	15.3	59.0	64.2	72.3	64.9

Table 23: Comparative analysis of GPT4, LLaMA 2, and LLaMA 3 performance on Task 4 for original and switch experiments (transposed table)

	Base	Ran	Neg	Quan	Ext
GPT4	91.1 ± 0.2	95.0 ± 0.8	97.2 ± 0	93.7 ± 1.7	82.9 ± 1.6
LLaMA2	64.4	45.6	25.2	8.4	59.2
LLaMA3	67.2	23.2	26.4	10	55.6
Human	100	98.0	100	100	100

Table 24: Accuracy comparison between GPT4, LLaMA2, LLaMA3 and human evaluation on Task 5; column title adopted from Table 11. Each cell reports the accuracy of *n* instances. *n* is 50 for human evaluation and 250 for models.

does X1 p'. A test instance is created by replacing the predicate 'p' in the template with a random string and assigning the random string one predicate type from Table 4. Each predicate type has 200 instances as a separate dataset. For the instances in all four datasets, the answer choices are all the same, namely '(a) Only X2, (b) Only X3, (c) Only X4, (d) X2 and X3, (e) X2 and X4, (f) X3 and X4, (g) X2, X3, and X4'. We intentionally keep the answer choices exactly the same across four datasets to eliminate the effect of ordering of answer choices on model performance. At the same time, depending on the predicate type, the gold answer will be different.

Next, we create a comparison set for each predicate type. In the comparison set, we use actual English words rather than random strings as predicate. Instead, the entities in the sentences are replaced by random strings to create 200 instances.

D Full prompts for Task 1

Table 26 shows the full prompts that we used for Task 1. The <Example> field in the template is empty for 0-shot; it is replaced by "<Example> (1-shot)" for 1-shot, "<Example> (1-shot-CoT)" for 1-shot-CoT, etc.

E Full GPT4 Output for Task 1 Case Study

In this appendix, we show the full answers produced by GPT4 on the example used in Task 1 case study, where the precedence is "+" = "*" > "-", and the test instance is 2 - 7 + 2 * 5.

E.1 0-shot output

Based on the new priorities where '+' has the same priority as '', and both are higher than '-', we would perform the operations in the following order:*

*1. Perform the multiplication and addition first (since they have the same priority, we'll perform them from left to right): 2 - 7 + (2 * 5)*

2. Calculate the multiplication: 2 - 7 + 10

3. Perform the addition: 2 + 10 = 12

4. Finally, perform the subtraction: 12 - 7 = 5

*So, the value of the expression 2 - 7 + 2 * 5 with the given priorities is [5].*

	NT & NS	NT & S	T & NS	T & S
GPT4	98.2 ± 1.5	87.8 ± 1.1	85.8 ± 2.8	48.3 ± 2.1
LLaMA2	2.0	11.0	14.0	31.0
LLaMA3	0	1.0	4.0	47.0
Human	100	100	100	100

Table 25: Accuracy comparison between GPT4, LLaMA2, LLaMA3, and human evaluation on Task 6; column title adopted from Table 12. Each cell reports the accuracy of *n* instances. *n* is 50 for human evaluation and 200 for models.

Template We redefine the priorities of elementary arithmetic operators as follows: <Priorities> <Example> Based on the new priorities, what is the value of <Test Instance>? You can give the result as a decimal, and please put the final numeric result in brackets [], for example, [1].
<Priorities> '+' = '/' > '*'
<Example> (1-shot) Example Question: What is the value of 62 * 19 + 57 / 74? Answer: [63.67567567567567].
<Example> (1-shot-CoT) Example Question: What is the value of 62 * 19 + 57 / 74? Answer: Based on the new rules, '+' and '/' have the same priority, which is higher than that of '*' Step1: Since '+' and '/' have the same priority over *, we evaluate '+' and '/' from left to right. In this case, '+' comes before '/', so we calculate '+' first: 19 + 57 = 76, so the expression becomes 62 * 76 / 74. Step2: '/' has priority over '*', now we calculate '/': 76 / 74 = 1.027027027027027, so the expression becomes 62 * 1.027027027027027. Step3: Finally, 62 * 1.027027027027027 = 63.67567567567567, so the final answer is [63.67567567567567].
<Test Instance> 98 * 43 + 29 / 25

Table 26: The prompts used for Task 1. The red, blue, and orange colors highlight the precedence assessments, the change of expressions, and the example instance, respectively, in the 1-shot-CoT setting.

E.2 0-shot-CoT output

Based on the new priorities you've given, the '+' operator (addition) has the same priority as the '' operator (multiplication), and both are higher than the '-' operator (subtraction). So we need to perform the addition and multiplication before the subtraction. Here's the expression with the new priorities:*

*2 - 7 + 2 * 5*

First, we perform the addition and multiplication since they have the same priority. We can do them from left to right:

7 + 2 = 9

*2 * 5 = 10*

Now we replace the original numbers with the results of these operations:

*2 - 9 * 10*

Next, we perform the multiplication, which has the same priority as addition but comes next in the sequence:

*9 * 10 = 90*

Now we replace the multiplication with its result:

2 - 90

Finally, we perform the subtraction:

2 - 90 = -88

*So the value of the expression 2 - 7 + 2 * 5 with the new priorities is [-88].*

E.3 1-shot output

Based on the new priorities where '+' has the same priority as '' and both are higher than '-', we should perform the addition and multiplication before the subtraction. So for the expression 2 - 7 + 2 * 5, we first do the addition and multiplication:*

*7 + 2 * 5 = 7 + (2 * 5) = 7 + 10 = 17*

Now we perform the subtraction:

ID	Move	AP	1D	2D	3D	4D	5D
(a) number of operators = 2							
1	no move	0-s	100	97.0	80.0	50.5	43.5
2		0-s-c	99.5	98.5	71.0	51.0	41.5
3		1-s	100	99.0	82.5	52.5	46.5
4		1-s-c	100	100	82.0	53.5	47.5
5	one move	0-s	94.5	91.5	80.0	57.0	45.5
6		0-s-c	97.0	96.0	82.0	53.0	44.5
7		1-s	89.5	93.5	74.0	55.0	43.5
8		1-s-c	100	100	88.5	65.5	57.0
(b) number of operators = 3							
9	no move	0-s	99.0	95.0	67.5	31.5	20.0
10		0-s-c	99.5	95.5	66.5	26.5	19.0
11		1-s	99.5	97.5	69.0	27.5	18.5
12		1-s-c	100	98.5	69.0	28.0	20.5
13	one move	0-s	75.0	58.5	40.5	17.5	14.5
14		0-s-c	74.5	57.0	32.5	11.5	11.0
15		1-s	67.0	58.5	36.0	17.5	12.0
16		1-s-c	94.5	97.5	73.0	35.5	32.5
17	two moves	0-s	54.0	39.5	32.5	15.5	13.5
18		0-s-c	46.0	42.0	24.0	10.0	7.0
19		1-s	41.0	36.0	25.5	13.5	11.5
20		1-s-c	92.0	94.0	79.5	48.5	46.0

Table 27: The accuracy of GPT4-Turbo on Task 1. "AP" denotes the approach, where "s" and "c" denote "*shot*" and "*CoT*", respectively (e.g., "1-s-c" means 1-shot-CoT). Col 1D-5D refers to the numbers of digits in operands, ranging from 1 to 5. Each cell reports the accuracy on 200 test instances. When the number of operators is fixed, the arithmetic expressions used in each column are the same, but the priorities of the operators in the expressions are different between no move, one move, and two moves.

2 - 17 = -15
*Therefore, the value of the expression 2 - 7 + 2 * 5 is [-15].*

E.4 1-shot-CoT output

Based on the new priorities where '+' and '' have the same priority and are higher than '-', we evaluate '+' and '*' before '-'. We also evaluate from left to right since '+' and '*' have the same priority.*

Step 1: Evaluate '+' before ''. So we calculate 7 + 2 first: 7 + 2 = 9, so the expression becomes 2 - 9 * 5.*

Step 2: Now we evaluate '', which has the same priority as '+', but since there's no '+' left, we just perform the multiplication: 9 * 5 = 45, so the expression becomes 2 - 45.*

Step 3: Finally, we perform the subtraction: 2 - 45 = -43.

ID	S	M	EM	BLEU	METEOR			F1
					Score	C	UA	
1	0	0	100	100	100	1.00	0.0	-
2	0	1	69.0	98.5	96.3	1.51	0.54	85.9
3	0	2	46.0	97.8	94.0	1.94	0.87	88.3
4	0	3	30.0	95.5	89.5	2.56	1.59	87.9
5	1	0	67.5	97.6	95.7	2.10	0.29	-
6	1	1	25.5	94.8	88.0	2.57	1.46	60.0
7	1	2	10.0	90.6	76.8	3.75	3.11	87.8
8	1	3	4.5	89.1	71.6	4.12	4.22	61.5
9	2	0	61.5	95.4	91.3	2.21	0.93	-
10	2	1	18.0	93.7	84.9	3.59	1.73	59.7
11	2	2	8.0	91.3	77.4	4.27	3.03	60.6
12	2	3	0.5	89.0	69.5	5.00	4.46	58.3
13	3	0	66.0	97.0	94.1	2.08	0.6	-
14	3	1	11.0	93.8	85.4	3.51	1.69	61.5
15	3	2	4.5	91.6	77.3	4.65	2.91	64.2
16	3	3	0.1	88.8	68.0	5.58	4.45	59.3

Table 28: The performance of GPT4 on Task 3 with 0-shot prompts. "S" and "M" represent the number of syntactic and morphological rules included in the prompts. "EM" is exact match, "F1" is Morph F1, and for METEOR, we report the METEOR score, the number of aligned chunks C and the number of unaligned target words UA.

So the final answer is [-43].

	EM	BLEU	METEOR			F1
			Score	C	UA	
(a) S=0 M=0						
0-s	100	100	100	1.00	0.0	-
(b) S=3 M=0						
0-s	66.0	97.0	94.1	2.08	0.6	-
0-s-c	60.0	99.9	98.3	2.81	0	-
1-s	77.0	99.9	99.3	1.90	0.01	-
1-s-c	93.0	99.8	99.8	1.27	0.01	-
(c) S=0 M=3						
0-s	30.0	95.5	89.5	2.56	1.59	87.9
0-s-c	27.0	96.3	90.1	2.33	1.62	86.8
1-s	54.0	98.4	96.9	1.46	0.57	94.3
1-s-c	49.0	98.6	95.6	1.51	0.64	93.1
(d) S=3 M=3						
0-s	1.0	88.8	68.0	5.58	4.45	59.3
0-s-c	1.5	90.8	75.6	5.44	3.86	62.4
1-s	19.5	98.0	92.3	3.45	1.06	90.3
1-s-c	33.5	97.3	93.4	2.10	1.11	88.9

Table 29: The performance of GPT4 on Task 3 with four types of prompts

F Example prompts for Task 2

In this appendix, we show the four types of prompts used in Task 2: For 0-shot, we provides the full prompt. For 1-shot-CoT, we only include the *[Example Question]* and *[Answer]*. The *[Example Question]* is just a multiple-choice question, and the *[Answer]* part is where the three types of 1-shot-CoT prompts differ.

F.1 0-shot prompt

*For the expression 8 - 4 * 4 + 9, the precedence of the operators is redefined. If 8 - 4 * 4 + 9 = 52, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*
A. '+' = '-' = ''*
B. '-' = '' > '+'*
C. '+' > '' = '-'*
D. '-' = '+' = ''*

F.2 1-shot-CoT-Simple

[Example Question]:
*For the expression 3 * 9 + 4 - 9, the precedence of the operators is redefined. If 3 * 9 + 4 - 9 = 22, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*
A. '' = '+' = '-'*
B. '+' = '-' > ''*
C. '-' > '+' > ''*
D. '+' > '' > '-'*

[Answer]:
*Using the precedence rule in the correct answer choice, the expression 3 * 9 + 4 - 9 should evaluate to 22. We test four options one by one.*
Step 1: using the precedence rule of choice A: '' = '+' = '-', the expression 3 * 9 + 4 - 9 evaluates to 22*
Step 2: using the precedence rule of choice B: '+' = '-' > '', the expression 3 * 9 + 4 - 9 evaluates to 12*
Step 3: using the precedence rule of choice C: '-' > '+' > '', the expression 3 * 9 + 4 - 9 evaluates to 12*
Step 4: using the precedence rule of choice D: '+' > '' > '-', the expression 3 * 9 + 4 - 9 evaluates to 30*
Given above, only with choice A, the expression evaluates to 22, so the correct answer should be [A]

*For the expression 8 - 4 * 4 + 9, the precedence of the operators is redefined. If 8 - 4 * 4 + 9 = 52, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*
A. '+' = '-' = ''*
B. '-' = '' > '+'*
C. '+' > '' = '-'*
D. '-' = '+' = ''*

F.3 1-shot-CoT-partial

[Example Question]:
*For the expression 3 * 9 + 4 - 9, the precedence of the operators is redefined. If 3 * 9 + 4 - 9 = 22, what is the redefined precedence of the operators? Please choose the correct answer from*

A, B, C, and D, and put the final answer choice in brackets, for example, [B].
A. '' = '+' = '-'*
B. '+' = '-' > ''*
C. '-' > '+' > ''*
D. '+' > '' > '-'*

[Answer]:
*Using the precedence rule in the correct answer choice, the expression 3 * 9 + 4 - 9 should evaluate to 22. We test four options one by one.*
*For A: The value of 3 * 9 + 4 - 9 can be calculated as follows: Based on the precedence rules of this answer choice '*' = '+' = '-', *, +, and - have the same priority,so we can evaluate the expression from left to right Step1: Firstly, we calculate '*': 3 * 9 = 27, so the expression becomes 27 + 4 - 9. Step2: Then we calculate '+': 27 + 4 = 31, so the expression becomes 31 - 9. Step3: Finally, 31 - 9 = 22, so this answer choice evaluates to 22.*

Similarly, we evaluate the expressions in B, C, and D in the same way as we evaluate the expression in A.
For B: using the precedence rule of choice B: '+' = '-' > '', the expression 3 * 9 + 4 - 9 evaluates to 12*
For C: using the precedence rule of choice C: '-' > '+' > '', the expression 3 * 9 + 4 - 9 evaluates to 12*
For D: using the precedence rule of choice D: '+' > '' > '-', the expression 3 * 9 + 4 - 9 evaluates to 30*
Given above, only with choice A, the expression evaluates to 22, so the correct answer should be [A]

*For the expression 8 - 4 * 4 + 9, the precedence of the operators is redefined. If 8 - 4 * 4 + 9 = 52, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*
A. '+' = '-' = ''*
B. '-' = '' > '+'*
C. '+' > '' = '-'*
D. '-' = '+' = ''*

F.4 1-shot-CoT-full

[Example Question]:
*For the expression 3 * 9 + 4 - 9, the precedence of the operators is redefined. If 3 * 9 + 4 - 9 = 22, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*
A. '' = '+' = '-'*
B. '+' = '-' > ''*
C. '-' > '+' > ''*
D. '+' > '' > '-'*

[Answer]:
*Using the precedence rule in the correct answer choice, the expression 3 * 9 + 4 - 9 should evaluate to 22. We test four options one by one.*
*For A: The value of 3 * 9 + 4 - 9 can be calculated as follows: Based on the precedence rules of this answer choice '*' = '+' = '-', *, +, and - have the same priority,so we can evaluate the expression from left to right Step1: Firstly, we calculate '*': 3 * 9 = 27, so the expression becomes 27 + 4 - 9. Step2: Then we calculate '+': 27 + 4 = 31, so the expression becomes 31 - 9. Step3: Finally, 31 - 9 = 22, so this answer choice evaluates to 22.*

*For B: The value of 3 * 9 + 4 - 9 can be calculated as follows: Based on the precedence rules of this answer choice '+' = '-' > '*', '+' and '-' have the same priority, which is higher than that of '*' Step1: Since '+' and '-' have the same priority over *, we evaluate '+' and '-' from left to right. In this case, '+' comes before '-', so we calculate '+' first: 9 + 4 = 13, so the expression becomes 3 * 13 - 9. Step2: '-' has priority over '*'. Now we calculate '-': 13 - 9 = 4, so the expression becomes 3 * 4. Step3: Finally, 3 * 4 = 12, so this answer choice evaluates to 12.*

*For C: The value of 3 * 9 + 4 - 9 can be calculated as follows: Based*

on the precedence rules of this answer choice '-' > '+' > '', '[0]' has the highest priority, followed by '+'. '*' has the lowest priority. Step1: Since '-' has the highest priority, we calculate '-' first: 4 - 9 = -5, so the expression becomes 3 * 9 + -5. Step2: Since '+' has priority over '*', we then calculate '+': 9 + -5 = 4, so the expression becomes 3 * 4. Step3: Finally, 3 * 4 = 12, so this answer choice evaluates to 12.*

*For D: The value of 3 * 9 + 4 - 9 can be calculated as follows: Based on the precedence rules of this answer choice '+' > '*' > '-', '[0]' has the highest priority, followed by '*'. '-' has the lowest priority. Step1: Since '+' has the highest priority, we calculate '+' first: 9 + 4 = 13, so the expression becomes 3 * 13 - 9. Step2: Since '*' has priority over '-', we then calculate '*': 3 * 13 = 39, so the expression becomes 39 - 9. Step3: Finally, 39 - 9 = 30, so this answer choice evaluates to 30.*

Given above, only with choice A, the expression evaluates to 22, so the correct answer should be [A]

*For the expression 8 - 4 * 4 + 9, the precedence of the operators is redefined. If 8 - 4 * 4 + 9 = 52, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*
A. '+' = '-' = ''*
B. '-' = '' > '+'*
C. '+' > '' = '-'*
D. '-' = '+' = ''*

G Example prompts for Task 3

In this appendix, we present the complete set of four distinct prompt types utilized in Task 3.

G.1 0-shot

The following are the linguistic rules of Language A.

In Language A, the canonical word order of sentences follow a Subject-Object-Verb structure.

Example: 'The cat eats the mouse' should be ordered as 'The cat the mouse eats'.

Prepositional phrases that modify a verb remain positioned after the verb.

Similarly,adverbs that modify a verb still maintain their position after the verb.

Prepositional phrases that modify a noun remain positioned after the noun.

In Language A, the morphological rule for nouns involves indicating a singular noun by adding 'S_' as a prefix to its lemma, and indicating a plural noun by adding 'P_' as a prefix to its lemma.

Example: 'These cats in the park' is expressed as 'These P_cat in the park' to represent plural and 'A cat in the park' is expressed as 'A S_cat in the park' to represent singular.

For anything not explicitly specified, Language A follows the same word order as English.

Here are the translated pairs of English lemmas into lemmas in language A: (the, tgt_the), (across, tgt_across), (field, tgt_field), (a, tgt_a), (farmer, tgt_farmer), (carefully, tgt_carefully), (bright, tgt_bright), (., .), (smile, tgt_smile), (basket, tgt_basket), (carry, tgt_carry), (heavy, tgt_heavy), (with, tgt_with) .

Translate the following simple sentence into Language A: T̈he farmer with a bright smile carries the heavy basket carefully across the field .̈

Please put the final translated result in brackets [] at the end of your response, for example, [tgt_the tgt_cat tgt_the tgt_dog tgt_chases].

G.2 0-shot-CoT

The following are the linguistic rules of Language A.

In Language A, the canonical word order of sentences follow a Subject-Object-Verb structure.

Example: 'The cat eats the mouse' should be ordered as 'The cat the mouse eats'.

Prepositional phrases that modify a verb remain positioned after the verb.

Similarly,adverbs that modify a verb still maintain their position after the verb.

Prepositional phrases that modify a noun remain positioned after the noun.

In Language A, the morphological rule for nouns involves indicating a singular noun by adding 'S_' as a prefix to its lemma, and indicating a plural noun by adding 'P_' as a prefix to its lemma.

Example: 'These cats in the park' is expressed as 'These P_cat in the park' to represent plural and 'A cat in the park' is expressed as 'A S_cat in the park' to represent singular.

For anything not explicitly specified, Language A follows the same word order as English.

Here are the translated pairs of English lemmas into lemmas in language A: (the, tgt_the), (across, tgt_across), (field, tgt_field), (a, tgt_a), (farmer, tgt_farmer), (carefully, tgt_carefully), (bright, tgt_bright), (., .), (smile, tgt_smile), (basket, tgt_basket), (carry, tgt_carry), (heavy, tgt_heavy), (with, tgt_with) .

Translate the following simple sentence into Language A: T̈he farmer with a bright smile carries the heavy basket carefully across the field .̈

Please put the final translated result in brackets [] at the end of your response, for example, [tgt_the tgt_cat tgt_the tgt_dog tgt_chases].

Please think step by step.

G.3 1-shot

The following are the linguistic rules of Language A.

In Language A, the canonical word order of sentences follow a Subject-Object-Verb structure.

Example: 'The cat eats the mouse' should be ordered as 'The cat the mouse eats'.

Prepositional phrases that modify a verb remain positioned after the verb.

Similarly,adverbs that modify a verb still maintain their position after the verb.

Prepositional phrases that modify a noun remain positioned after the noun.

In Language A, the morphological rule for nouns involves indicating a singular noun by adding 'S_' as a prefix to its lemma, and indicating a plural noun by adding 'P_' as a prefix to its lemma.

Example: 'These cats in the park' is expressed as 'These P_cat in the park' to represent plural and 'A cat in the park' is expressed as 'A S_cat in the park' to represent singular.. Prepositional phrases that modify a verb remain positioned after the verb.

Adverbs that modify a verb remain positioned after the verb.

For anything not explicitly specified, Language A follows the same word order as English.

Here's an example that demonstrates how to translate an English sentence into Language A.

Example sentence: The black cat with a small tail eats the food quietly in the room .

Below, you'll find the translated pairs of English lemmas into lemmas in language A for the example sentence: [('the', 'tgt_the'), ('black', 'tgt_black'), ('cat', 'tgt_cat'), ('with', 'tgt_with'), ('a', 'tgt_a'), ('small', 'tgt_small'), ('tail', 'tgt_tail'), ('eats', 'tgt_eats'), ('food', 'tgt_food'), ('quietly', 'tgt_quietly'), ('in', 'tgt_in'), ('room', 'tgt_room'), ('.', '.')] Translated example sentence: tgt_the tgt_black S_tgt_cat tgt_with tgt_a tgt_small S_tgt_tail tgt_the S_tgt_food tgt_eats tgt_quietly tgt_in tgt_the S_tgt_room .

Here are the translated pairs of English lemmas into lemmas in language

A: (the, tgt_the), (across, tgt_across), (field, tgt_field), (a, tgt_a), (farmer, tgt_farmer), (carefully, tgt_carefully), (bright, tgt_bright), (., .), (smile, tgt_smile), (basket, tgt_basket), (carry, tgt_carry), (heavy, tgt_heavy), (with, tgt_with) .

Translate the following simple sentence into Language A: "The farmer with a bright smile carries the heavy basket carefully across the field .".

Please put the final translated result in brackets [] at the end of your response, for example, [tgt_the tgt_cat tgt_the tgt_dog tgt_chases].

G.4 1-shot-CoT

The following are the linguistic rules of Language A.

In Language A, the canonical word order of sentences follow a Subject-Object-Verb structure.

Example: 'The cat eats the mouse' should be ordered as 'The cat the mouse eats'.

Prepositional phrases that modify a verb remain positioned after the verb.

Similarly,adverbs that modify a verb still maintain their position after the verb.

Prepositional phrases that modify a noun remain positioned after the noun.

In Language A, the morphological rule for nouns involves indicating a singular noun by adding 'S_' as a prefix to its lemma, and indicating a plural noun by adding 'P_' as a prefix to its lemma.

Example: 'These cats in the park' is expressed as 'These P_cat in the park' to represent plural and 'A cat in the park' is expressed as 'A S_cat in the park' to represent singular.. Prepositional phrases that modify a verb remain positioned after the verb.

Adverbs that modify a verb remain positioned after the verb.

For anything not explicitly specified, Language A follows the same word order as English.

Here's an example that demonstrates how to translate an English sentence into Language A step by step.

Example sentence: The black cat with a small tail eats the food quietly in the room .

Below, you'll find the translated pairs of English lemmas into lemmas in language A for the example sentence: [('the', 'tgt_the'), ('black', 'tgt_black'), ('cat', 'tgt_cat'), ('with', 'tgt_with'), ('a', 'tgt_a'), ('small', 'tgt_small'), ('tail', 'tgt_tail'), ('eats', 'tgt_eats'), ('food', 'tgt_food'), ('quietly', 'tgt_quietly'), ('in', 'tgt_in'), ('room', 'tgt_room'), ('.', '.')]

To change the word order of the sentence to Subject-Object-Verb, we first identify the root verb, which is 'eats'.

Next, we find the subject of the sentence, 'cat', and its related tokens: ['The', 'black', 'with', 'a', 'small', 'tail']. Then, we locate the object of the sentence, 'food', and its related tokens: ['the'].

We then move the object and its related tokens behind the subject and its related tokens.

Now, the sentence is in Subject-Object-Verb word order as applied in Language A: 'The black cat with a small tail the food eats quietly in the room .'.

We translate the English sentence into Language A using the lemma pairs between English and Language A:

tgt_the tgt_black tgt_cat tgt_with tgt_a tgt_small tgt_tail tgt_the tgt_food tgt_eat tgt_quietly tgt_in tgt_the tgt_room . To apply the morphological rule for nouns, add 'S_' as a prefix for singular nouns and 'P_' as a prefix for plural nouns in their lemmas:

First identify singular nouns: tgt_cat, tgt_tail, tgt_food, tgt_room.

Then add 'S_' as a prefix to each singular noun: S_tgt_cat, S_tgt_tail, S_tgt_food, S_tgt_room

There is no plural noun in the sentence.

Now, the sentence follows the singular and plural noun rule in Language A: 'tgt_the tgt_black S_tgt_cat tgt_with tgt_a tgt_small S_tgt_tail tgt_the S_tgt_food tgt_eat tgt_quietly tgt_in tgt_the S_tgt_room .'.

Lastly, we have the translated

example sentence: tgt_the tgt_black S_tgt_cat tgt_with tgt_a tgt_small S_tgt_tail tgt_the S_tgt_food tgt_eats tgt_quietly tgt_in tgt_the S_tgt_room .

Here are the word pairs between English and the language A:Here are the translated pairs of English lemmas into lemmas in language A: (the, tgt_the), (across, tgt_across), (field, tgt_field), (a, tgt_a), (farmer, tgt_farmer), (carefully, tgt_carefully), (bright, tgt_bright), (., .), (smile, tgt_smile), (basket, tgt_basket), (carry, tgt_carry), (heavy, tgt_heavy), (with, tgt_with) .

Translate the following simple sentence into Language A: "The farmer with a bright smile carries the heavy basket carefully across the field .".

Please put the final translated result in brackets [] at the end of your response, for example, [tgt_the tgt_cat tgt_the tgt_dog tgt_chases].