

## A APPENDIX ORGANIZATION

Our anonymized code is provided with our paper submission. Due to space limitations (our complete results are over 4GB), we have provided two of the batches of the zero-shot prompting results and FOLIO and LogiEval datasets.

The appendix is organized as follows. Appendix B provides the algorithm used for dataset generation. Appendix C discusses prompt tuning and validating our prompts on 3SAT. Appendix D provides the parameters we used when generating the five datasets discussed in the paper. Appendix E provides additional information on our experimental setup, including the computational resources used. Appendix F discusses the prompts and provides examples. Appendix G is our detailed analysis of the empirical results from the main paper. Appendix H discusses an experiment we ran to evaluate the standard deviation error. Appendix I includes additional results from our zero-shot prompting experiments using other metrics for categorization. Appendix J evaluates an experiment we performed comparing few-shot prompting compared to zero-shot. Finally, Appendix K provides the experimental setup of the benchmarks we evaluated, data values and sources of scores collected, the  $\forall\text{uto}\exists\forall\text{L}$  scores used for comparison, and additional correlation results.

## B DATASET GENERATION

In this section, we provide the algorithm for generating formal syntax (FS) expressions and show that it can generate all possible expressions from the grammar and vocabulary.

Our approach,  $\forall\text{uto}\exists\forall\text{L}$ , generates datasets by constructing a context-free grammar (CFG) tree using the grammars discussed in Section 4. Since it is intractable to generate the full tree, we control the branching factor and randomly expand the branches of this tree to generate formulae.

---

### Algorithm 1 Dataset Generation

---

```

1: Inputs: CFG  $\mathcal{G}$ , vocabulary  $\mathcal{V}$ , branching factor  $n$ , tree depth  $depth$ , sample count  $sample\_count$ ,
   and categorization metric  $m$ .
2: Outputs: set of FS expressions  $\bar{\varphi}$ 
3:  $\mathcal{N} \leftarrow \{0 : [None]\}, \mathcal{N}_t \leftarrow \langle \rangle$ 
4: for  $d = 1, 2, \dots, depth$  do
5:    $\mathcal{N}' \leftarrow sampleN(\mathcal{N}[d-1], n)$ 
6:   for  $\nu \in \mathcal{N}'$  do
7:      $\mathcal{N}_\nu, \mathcal{T}_\nu \leftarrow generateNChildren(\nu, \mathcal{G}, n)$ 
8:      $\mathcal{N}[d] += \mathcal{N}_\nu$ 
9:      $\mathcal{N}_t \leftarrow \mathcal{N}_t \cup \mathcal{T}_\nu$ 
10:  end for
11: end for
12:  $M \leftarrow categorizeExpressionsIntoDict(\mathcal{N}_t, m)$ 
13:  $\bar{\varphi} \leftarrow \langle \rangle$ 
14: for  $k \in keys(M)$  do
15:    $M_k \leftarrow sampleCFGExpressions(M[k], sample\_count)$ 
16:    $\bar{\varphi}_k \leftarrow buildFSExpressions(M_k, \mathcal{V})$ 
17:    $\bar{\varphi} \leftarrow \bar{\varphi} \cup \bar{\varphi}_k$ 
18: end for
19: Return:  $\bar{\varphi}$ 

```

---

The dataset generation algorithm is shown in Algorithm 1. This algorithm constructs a CFG tree by maintaining non-terminal nodes at each tree level ( $\mathcal{N}$ ) and all the leaf nodes ( $\mathcal{N}_t$ ), where each terminal node represents a completed CFG expression (line 3). For generating nodes at a certain level in the tree,  $n$  nodes from the previous level are sampled (line 5). Each node is branched  $n$  times using the CFG to produce nodes at the current tree level, and all the leaf nodes are collected (lines 7 through 9). As a result, by iteratively performing this process for each tree level, we obtain a set of leaf nodes (CFG expressions).

The leaf nodes are then categorized based on the specified metric (e.g., tree depth, number of operators, etc.) (line 12). For each metric value, a fixed number of CFG expressions corresponding to that

value are sampled (line 15). Using the vocabulary, an FS expression is constructed from each CFG expression (line 16). Consequently, the final dataset of FS expressions contains an equal number for each metric value (line 17). This set of FS expressions is the final result produced by the algorithm (line 19).

The vocabulary is fixed in length, with a hyperparameter controlling the number of unique propositions for propositional logic. Similarly, for first-order logic, the number of unique variables, constants, and predicates are also hyperparameters. Also, regular expressions have a hyperparameter controlling the alphabet size. When these expression components are needed for building the FS expression, the exact one is selected using uniform random selection. In the special case of first-order logic predicates, the grounded predicate is generated by randomly selecting a predicate and then selecting constants depending on the predicate’s arity. In the case of the arbitrary vocabulary, the arity for a predicate is randomly assigned. To add variables, each constant has a certain probability of being replaced by a variable.

**Guaranteed Expression Coverage** The dataset generator (Algorithm 1) is guaranteed to generate all possible formal syntax expressions that can be produced for a grammar and vocabulary. Let  $\varphi$  be an FS expression that can be constructed using the rules from CFG  $\mathcal{G}$  and the vocabulary  $\mathcal{V}$ . Note that  $\varphi$  corresponds to a CFG expression  $\varphi_{CFG}$ , derived by substituting the vocabulary with the CFG symbols. Due to uniform selection, the probability of  $\varphi$  being generated from  $\varphi_{CFG}$  is greater than zero. Furthermore, the CFG expression represents a leaf node in the CFG tree that can be reached by applying the CFG rules in a specific sequence. Due to the random sampling of rules at each node, there is a non-zero probability of generating this particular path in the tree. Thus,  $\varphi$  can be generated using the dataset generator algorithm.

### C 3-SAT PROMPT CALIBRATION

In this section, we discuss the KSAT results used to calibration the prompts.

We tested several prompts for 3-SAT to verify that our prompts are sufficient to prompt the LLM to correctly perform informalization and autoformalization. Additionally, we verified that the equivalence verification prompt prompted the LLMs to give an accurate yes-or-no answer. The performance of all six LLMs on 3-SAT for §A2, §A3, and §A4 are shown in Figure 7.

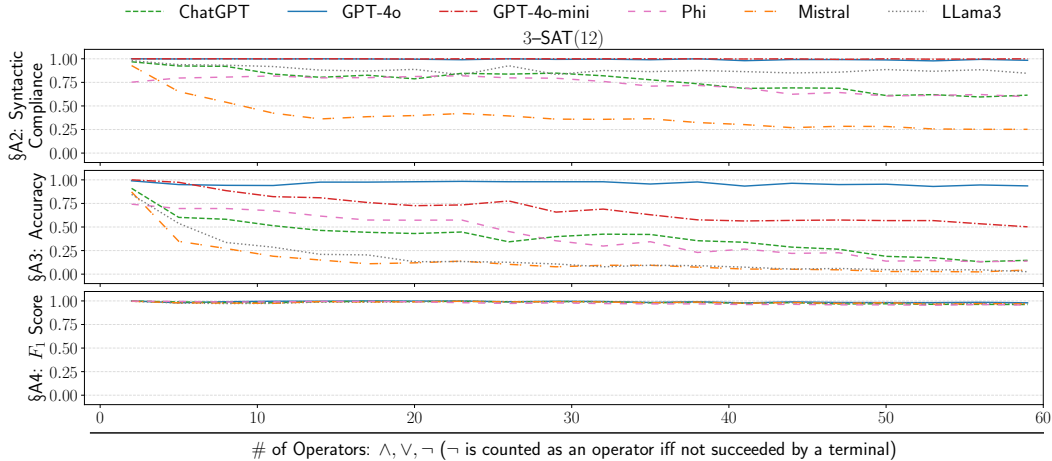


Figure 7: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) for 3-SAT from using  $\forall\text{to}\exists\forall\text{L}$  to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the # of operators.

The best-performing models we tested (GPT-4o and GPT-4o-mini) achieved nearly perfect syntactic compliance, accuracy, and equivalence verification even as the number of operators increased. This

proves that the prompts we used in our experiments are sufficient for prompting the model for performing the tasks for §A2, §A3, and §A4.

For the other LLMs tested, syntactic compliance and accuracy diminished as the number of operators increased. However, when evaluating the equivalence of GPT-4o results, all LLMs achieved near-perfect accuracy regardless of operator number. Due to most of GPT-4o results being positive cases, the results support that LLMs can verify two equivalent 3-SAT formulae as equivalent.

## D DATASET GENERATION HYPERPARAMETERS

In Table 1, we provide the hyperparameters used to generate the five datasets.

Table 1: Hyperparameters used for producing the five datasets.

Parameter Type	Hyperparameter	Value	Description
<b>General</b>	depth	40	Maximum depth of the CFG tree.
	n	200	Branching factor of produced CFG tree.
	sample_count	50	Number of CFGS for each metric value to select.
<b>First-Order Logic</b>	free_variable_prob	0.25	Probability of a constant being replaced by a variable.
	max_free_variables	$\infty$	Maximum number of unique variables.
	max_predicate_arity	2	Maximum predicate arity.
	min_predicate_arity	1	Minimum predicate arity.
	num_objects	12	Number of unique constants.
	num_predicates	8	Number of unique predicates.
<b>Propositional Logic</b>	num_propositions	12	Number of unique propositions.
<b>Regular Expression</b>	alphabet_size	2	Alphabet size.

## E EXPERIMENTAL SETUP

In this section, we will provide the details of our experimental setup for generating the datasets and running  $\forall$ uto $\exists$ \forall L for evaluating each LLM’s performance.

We ran our experiments using Python 3.10.13 with package versions shown in Table 2. We also repackaged Prover9 (McCune, 2010) to improve performance where this repackaged version can be found in our code base.

**Dataset Generation:** We generated five datasets using the dataset generation algorithm with the hyperparameters shown in Table 1 using the number of operators as the categorization metric for all but regular expression, where we used CFG tree depth. We generated 10 batches for each dataset, resulting in approximately 20k samples for each dataset with an equal distribution for each operator number.

**Evaluating and Verification:** The closed-source models (GPT-3.5-turbo, GPT-4o, and GPT-4o-mini) were accessed using their API using a temperature of 0.1. The open-source models Llama-3-8B-Instruct and Mistral-v0.2-7B-Instruct were locally hosted on a server with a 13th Gen Intel(R) Core(TM) i9-13900K and Nvidia RTX 4090 GPU using the model’s default parameters with a temperature of 1. Similarly, Phi-3-medium-4k-instruct was locally hosted on a server using a Nvidia A100-XM4-80GB GPU.

Table 2: Python package versions used for empirical evaluation.

Python Package	Version
openai	1.45.0
nltk	3.8.1
tqdm	4.66.4
anthropic	0.26.1
backoff	2.2.1
tiktoken	0.6.0
transformers	4.41.1
Faker	25.2.0
networkx	3.3

### Prompt 2: Few-Shot First-Order Logic Informalization Prompt

#### [TASK]

Your task is to convert a first-order logic formula, appearing after [FORMULA], to a natural description that represents the formula. Only natural language terms are allowed to be used and do not copy the formula in your description. Your description should allow one to reconstruct the formula without having access to it, so make sure to use the correct names in your description. Explicitly describe the predicates. You may use terms verbatim as specified in the vocabulary below.

#### [EXAMPLE 1]

$(\neg p2 \vee p1 \vee \neg p2)$

Disjunctive predicate logic expression consisting of three components: the negation of a proposition labeled p2, the proposition p1, and again the negation of p2.

#### [EXAMPLE 2]

$(\neg \neg p2 \wedge \neg(p3 \vee p1))$

The expression asserts that p2 is not false while both p3 and p1 are not true.

#### [VOCABULARY]

$\vee$  represents disjunction

$\wedge$  represents conjunction

$\neg$  represents negation

( and ) represent parentheses

propositions can be used verbatim

predicates can be used verbatim

$\forall < x1 > < x2 > \dots < xn > .$  represents universal quantification with  $x1 \dots$  representing free variables

$\exists < x1 > < x2 > \dots < xn > .$  represents existential quantification with  $x1 \dots$  representing free variables

The objects are:  $p5, x1$

The parameterized predicates are:  $pred3(?p0, ?p1)$

The free variables are:  $x1$

#### [FORMULA]

$\forall x1 \ pred3(p5, x1)$

Verification was performed on an AMD EPYC machine with 128 cores.

## F PROMPTING

In this section, we provide the zero-shot and few-shot used in the main paper experiments.

The prompt for each dataset type provides the LLM with information on the problem type and the vocabulary. For informalization, we prompt the model to produce just a natural language description. We also provide the list of objects, predicates, propositions, and free variables in the formal syntax expression. For autoformalization, the LLM is prompted to provide just the formal syntax expression using the natural language description. Additionally, for first-order logic with a non-synthetic grammar, we provide the predicate names and arity in the autoformalization prompt. Two examples are provided for few-shot prompting.

For §A4, the prompt used for using an LLM to verify the equivalence of two formulae tells the LLM about the type of datasets (e.g., propositional logic, first-order logic, and regular expression). Using Chain-of-Thought prompting, the model is prompted to provide an explanation before giving a yes-or-no answer in a parsable format. Below are examples of the exact prompts used.

## Prompt 3: Few-Shot First-Order Logic Autoformalization Prompt

## [VOCABULARY]

Use  $\vee$  to represent disjunction

Use  $\wedge$  to represent conjunction

Use  $\neg$  to represent negation

Use ( and ) to represent parentheses

Use  $\forall$  <free\_variable\_list> to represent universal quantification

Use  $\exists$  <free\_variable\_list> to represent existential quantification

The <free\_variable\_list> consists of a sequence of space separate free variables with the last variable immediately followed by a period. Examples: (1) all x1 x2. (2) exists x4.

Use <predicate>(<parameter\_list>) to represent predicates (Names and parameters are provided in the description)

## [TASK]

Your task is to interpret the natural language (NL) description of a first-order logic formula and represent it as formal syntax using the vocabulary specified in the [VOCABULARY] block above. Only output the formula and no other text. The NL description appears immediately following the [NL DESCRIPTION] tag.

## [EXAMPLE 1]

Disjunctive predicate logic expression consisting of three components: the negation of a proposition labeled p2, the proposition p1, and again the negation of p2.

$(\neg p2 \vee p1 \vee \neg p2)$

## [EXAMPLE 2]

The expression asserts that p2 is not false while both p3 and p1 are not true.

$(\neg \neg p2 \wedge \neg (p3 \vee p1))$

## [NL DESCRIPTION]

For all objects labeled x1, the predicate pred3 holds true with parameters p5 and x1.

## G ANALYSIS OF MAIN PAPER RESULTS

In this section, we analyze the main empirical results of the paper. Our results clearly show that current SOTA LLMs are not performant in the truth maintenance task, which is why  $\forall\text{uto}\exists\forall\wedge\text{L}$  is needed. As the expression complexity increases, the syntactic compliance, accuracy, and ability to verify equivalence diminishes. We describe some of the errors that cause the low accuracy for propositional logic, first-order logic, and regular expressions.

## G.1 PROPOSITIONAL LOGIC RESULTS

**Informalization Errors:** A common error was the LLM failed to describe the proposition names. Another was the LLM failing to provide a complete description of the formula. For example, GPT-3.5-turbo often described portions of the expression based on what propositions and operators it contained. A common issue with GPT-4o, one of the best models, is that it often uses different propositional symbols (see example 5 in Table 3). Finally, we also observed hallucinations where the LLM attempted and failed to simplify the original formula (see example 4 in Table 3). These interpretation errors resulted in the original meaning of the expression being lost.

**Autoformalization Errors:** We observed there were often syntactic issues where the description was not fully translated into a formula or the parentheses did not match. An interesting result is that the LLMs struggled to place the negation operator in the correct location. For example, GPT-4o often describes  $\neg p \wedge \neg p$  as predicate  $p$  "negated twice and combined" but failed to regenerate the original formula properly with this description.

#### Prompt 4: Few-Shot Regex Informalization Prompt

[TASK]

Your task is to convert the regular expression appear after [REGEX], to a natural description that represents the regular expression. Only natural language terms are allowed to be used and do not copy the regular expression in your description. Your description should allow one to reconstruct the regular expression without having access to it, so make sure to use the correctly account for scoping. You may use terms verbatim as specified in the vocabulary below.

[VOCABULARY]

you may use symbols from the vocabulary

you can use \*

[EXAMPLE 1]

(1\*)0\*

The regex matches strings that starts with any number (including none) of the digit '1', followed by any number (including none) of the digit '0'.

[EXAMPLE 2]

(01\*)

The regex matches strings that begin with a '0' followed directly by any number (including none) of '1's.

[FORMULA]

0

#### Prompt 5:

Few-Shot Regex Autoformalization Formal [VOCABULARY]

Use \* to represent zero or more duplications of the same expression

Use ( and ) to represent parentheses

[TASK]

Your task is to interpret the natural language (NL) description of a regular expression and represent it as formal syntax using the vocabulary specified in the [VOCABULARY] block above. Only output the regular expression and no other text. The NL description appears immediately following the [NL DESCRIPTION] tag.

[EXAMPLE 1]

The regex matches strings that starts with any number (including none) of the digit '1', followed by any number (including none) of the digit '0'.

(1\*)0\*

[EXAMPLE 2]

The regex matches strings that begin with a '0' followed directly by any number (including none) of '1's.

(01\*)

[NL DESCRIPTION]

The regex matches strings that start with the digit '0'.

## G.2 FIRST-ORDER LOGIC RESULTS

**Informalization Errors:** Similar to propositional logic, we observed the LLM often failed providing enough details resulting in incorrect formulas being generated. A significant source of errors we

#### Prompt 6: Zero-Shot Propositional Logic Informalization Prompt

[TASK]

Your task is to convert a propositional logic formula, appearing after [FORMULA], to a natural description that represents the formula. Only natural language terms are allowed to be used and do not copy the formula in your description. Your description should allow one to reconstruct the formula without having access to it, so make sure to use the correct names in your description. Explicitly describe the predicates. You may use terms verbatim as specified in the vocabulary below.

[VOCABULARY]

$\vee$  represents disjunction

$\wedge$  represents conjunction

$\neg$  represents negation

( and ) represent parentheses

propositions can be used verbatim

The propositions are: p5, p12, p4

[FORMULA]

$(p5 \vee \neg p12 \vee \neg p4)$

#### Prompt 7: Zero-Shot Propositional Logic Autoformalization Prompt

[TASK]

Your task is to interpret the natural language (NL) description of a propositional logic formula and represent it as formal syntax using the vocabulary specified in the [VOCABULARY] block above. Only output the formula and no other text. The NL description appears immediately following the [NL DESCRIPTION] tag.

[VOCABULARY]

Use  $\vee$  to represent disjunction

Use  $\wedge$  to represent conjunction

Use  $\neg$  to represent negation

Use ( and ) to represent parentheses

[NL DESCRIPTION]

A disjunctive statement involving three propositions: p5, the negation of p12, and the negation of p4.

observed when not providing the predicate names and arity was the LLM rephrasing its explanation causing confusion when regenerating.

**Autoformalization Errors:** Beyond the errors observed in propositional logic, the most common mistake made during autoformalization was the LLM confusing constants with variables (see example 2 in Table 4). Additionally, the LLMs often messed up the predicate arity. Mistral often used  $=$  and  $\neq$  operators with the variables, which was not needed for any formulae in  $\forall\text{to}\exists\forall\wedge\vee$ . Similarly, the LLMs would often use their own grammar instead of the one provided in the prompt.

### G.3 REGULAR EXPRESSION RESULTS

**Informalization Errors:** Most of the errors observed were the LLMs giving the wrong explanation, even for simple regular expressions. For example, GPT-4o often described  $c^*$  as "one or more occurrences of 'c'", where  $c$  is a character from the alphabet (see example 1 in Table 5). For the other LLMs, it was quite common for the explanation to not give the actual character (see example 4 in Table 5). Overall, we observed a higher likelihood of SOTA LLMs hallucinating on regular expressions compared to the other datasets.

## Prompt 8: First-Order Logic Verification Prompt

Your task is to say whether two First-Order Logic formulae are equivalent. The first formula will appear right after [FORMULA 1] and the second after [FORMULA 2].  
Give an explanation followed by a yes or no answer. The answer must show up at the end with the format "[Answer]" followed by either a yes or no.

[Formula 1]

$\exists x1. \neg pred5(p7)$

[Formula 2]

$\exists p7. \neg pred5(p7)$

Table 3: Examples of errors the evaluated LLMs made while evaluating  $\forall\text{uto}\exists\forall\wedge\text{L}$  for the propositional logic dataset.

$\varphi$	$\mathcal{I}(\varphi)$	$(\mathcal{A} \circ \mathcal{I})(\varphi)$
$(\neg p11 \wedge \neg p8)$	The natural description of the formula is: "It is not the case that both proposition p11 and proposition p8 are true."	$(\neg(p11 \wedge p8))$
$(\neg p10 \wedge (\neg p9 \vee p7))$	The natural description of the formula is: "It is not the case that proposition p10 is true and either proposition p9 is false or proposition p7 is true."	$\neg(p10 \wedge (\neg p9 \vee p7))$
$(\neg p3 \wedge \neg p7)$	Here is the natural description of the given formula: "The statement says that p3 is not true and p7 is also not true."	$(\neg p3 \vee \neg p7)$
$(\neg\neg p2 \vee p3)$	The formula states that either the proposition p2 is true or the proposition p3 is true. Additionally, it specifies that the proposition p2 is not false.	$(p2 \vee p3) \wedge \neg\neg p2$
$(\neg p2 \wedge p5 \wedge \neg p6)$	The formula describes a situation where the first proposition is false, the second proposition is true, and the third proposition is false.	$(\neg p \wedge q \wedge \neg r)$

**Autoformalization Errors:** The most common mistake when constructing a regular expression from natural language was misplacing  $*$  or adding it when it was not needed (see example 3 in Table 5). Finally, even though we explicitly prompted the LLMs to use just  $*$ , sometimes the LLM would use  $+$ .

## H STANDARD DEVIATION EVALUATION

In this section, we perform an empirical analysis of the standard deviation of the syntactic compliance and accuracy of the  $\forall\text{uto}\exists\forall\wedge\text{L}$  results. Due to the 10 batches having different data, the standard deviation cannot be computed reliably based on the performance of the individual batches. We evaluated the standard deviation by running  $\forall\text{uto}\exists\forall\wedge\text{L}$  10 times on the first batch of each dataset composed of 1974 propositional logic, 1900 first-order logic, and 1842 regular expressions examples. Additionally, we evaluated GPT-3.5-turbo (ChatGPT) with a temperature of 1, LLama-3-8B-Instruct, Mistral-v0.2-7B-Instruct, and Phi-3-medium-4k-instruct. We calculated the mean and standard deviation of each independent run of  $\forall\text{uto}\exists\forall\wedge\text{L}$  and plotted the results in Figure 8.

For propositional and first-order logic, the standard deviation of the evaluated LLMs is low. While noisier, the standard deviation of the regular expression results were still less than 20% with the better performing models having a lower standard deviation. Overall, this experiment shows that the



Table 4: Examples of errors the evaluated LLMs made while evaluating  $\forall\text{uto}\exists\forall\wedge\text{L}$  for the first-order logic dataset.

$\varphi$	$\mathcal{I}(\varphi)$	$(\mathcal{A} \circ \mathcal{I})(\varphi)$
$(\neg\text{pred8}(p10) \wedge \text{pred8}(p5) \wedge \text{pred6}(p8))$	The formula states that it is not the case that the predicate pred8 holds for the object p10, and the predicate pred8 holds for the object p5, and the predicate pred6 holds for the object p8, all conjoined together.	$\neg(\text{pred8}(p10) \wedge \text{pred8}(p5) \wedge \text{pred6}(p8))$
$\exists x1. \neg\text{pred2}(p4)$	Here is the natural description of the given formula: "There exists an x1 such that p4 is not an instance of pred2."	$\exists x1. \neg\text{pred2}(x1)$
$\forall x1. \neg\neg\text{pred3}(p5)$	For all x1, it is not the case that pred3(p5) is false.	$\forall x1. \neg(\text{pred3}(p5) \vee \neg\text{pred3}(p5))$
$\text{pred2}(p3, p5)$	There exists a relationship between the objects p3 and p5 such that the predicate pred2 holds true for these objects.	$\exists p3\ p5. \text{pred2}(p3, p5)$

Table 5: Examples of errors the evaluated LLMs made while evaluating with  $\forall\text{uto}\exists\forall\wedge\text{L}$  for the regular expression dataset.

$\varphi$	$\mathcal{I}(\varphi)$	$(\mathcal{A} \circ \mathcal{I})(\varphi)$
$1*0$	The pattern consists of one or more occurrences of the digit "1" followed by exactly one occurrence of the digit "0".	$(1^*)10$
$(1^*)^*0$	The regular expression describes a sequence that consists of zero or more repetitions of the digit '1', followed by exactly one '0'. This entire pattern can repeat any number of times, including zero times.	$((1^*)0)^*$
$1*11^*$	The pattern consists of zero or more occurrences of the digit '1', followed by exactly one occurrence of the digit '1', followed by zero or more occurrences of the digit '1'.	$1^*11^*$
$(1)^*0$	Zero or more occurrences of the character or group of characters before the asterisk.	$(. ^*)$

noise of non-deterministic text generation does not significantly impact  $\forall\text{uto}\exists\forall\wedge\text{L}$  or our results and evaluations.

## I ADDITIONAL ZERO-SHOT PROMPTING RESULTS

In this section, we evaluate other categorization metrics from the zero-shot prompting experiments from the main paper. For the propositional and first-order logic datasets, the other categorization metrics are the CFG parse tree depth needed to produce each FS expression and the individual number of each operator ( $\wedge$ ,  $\vee$ ,  $\neg$ ). For regular expressions, we have discussed in the main paper that each regular expression represents a minimal DFA that is unique up to isomorphism. Therefore, the other categorization metrics for regular expressions are the number of nodes  $V$ , the number of edges  $E$ , and the density of this minimal DFA. The density is calculated using Equation 1 where we discretize the value by rounding to every tenth.

$$\text{Density} = \frac{|E|}{|V|(|V| - 1)} \quad (1)$$

**Imbalanced Dataset Labels** Due to the datasets being created by sampling an equal number of expressions for each number of operators, taking this dataset and evaluating it in terms of the other

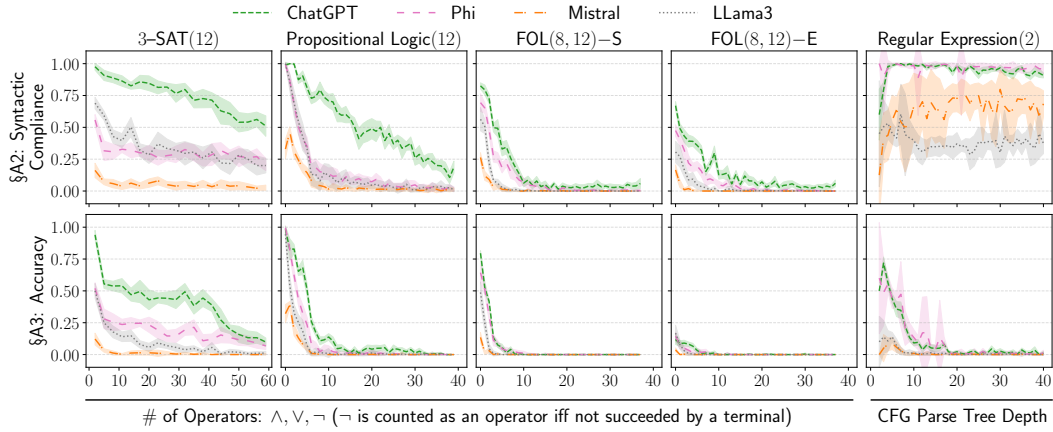


Figure 8: Average and standard deviation error of Zero-shot Pass@1 results from using  $\forall\text{auto}\exists\forall\wedge\vee\neg$  to assess LLMs w.r.t. §A2 and §A3 (Sec. 4) on the first batch of the packaged datasets. The x-axis represents an increasing order of descriptive complexity.

metrics results in an imbalanced dataset. To examine this effect, we have created Figures 9 and 10 to perform an analysis of dataset imbalance on these other metrics.

For propositional and first-order logic, the dataset is actually quite balanced due to CFG tree depth and the number of each individual operator having a high correlation to the total number of operators. As such, other than metric values close to the extrema, the noise from the imbalanced data will be marginal.

The regular expression dataset is less balanced due to a weaker correlation with the CFG tree depth. The middle of the density graphs will be the most noisy since there is significantly less data for densities of 0.1 and 0.2. The number of examples drops as the number of edges and nodes increases with less than 10% of the data having more than 7 edges and/or nodes.

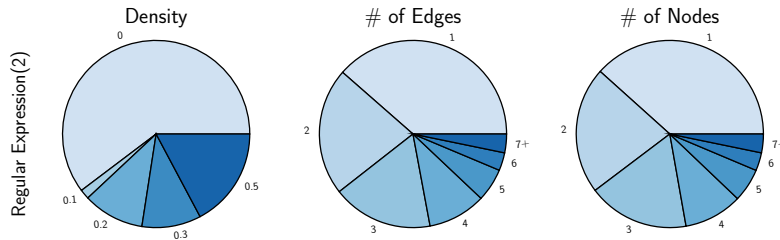


Figure 9: Count of the number of examples for each metric value for the regular expression datasets. The pie charts increase in values counter-clockwise while going from lighter to darker.

**Categorization Metrics Performance** In Figures 11, 12, 13, 14, and 15 the performance of each LLM over these other categorization metrics are shown. Across the board, we observe a diminishing performance regardless of the source of increasing complexity. Ignoring the noise from the low number of examples closer to the extrema, the depth of the tree showed a similar behavior as the operator number. Propositional logic performance was concave w.r.t the number of  $\wedge$  and  $\vee$  operators since it becomes easier to describe expressions composed of exclusively  $\wedge$  and  $\vee$  operators. A similar, but weaker pattern is observed in the first-order logic results for the same reason. The negation operator was not concave, showing how LLMs struggle to handle multiple negation operators.

For regular expressions, increasing the number of nodes and edges reduces accuracy and the ability to evaluate equality. Density does not seem to be a factor, as the dip at 0.1 can be associated with noise due to the lower number of examples. Overall, these three metrics are much weaker factors in how well the LLM performs compared to the CFG tree depth.

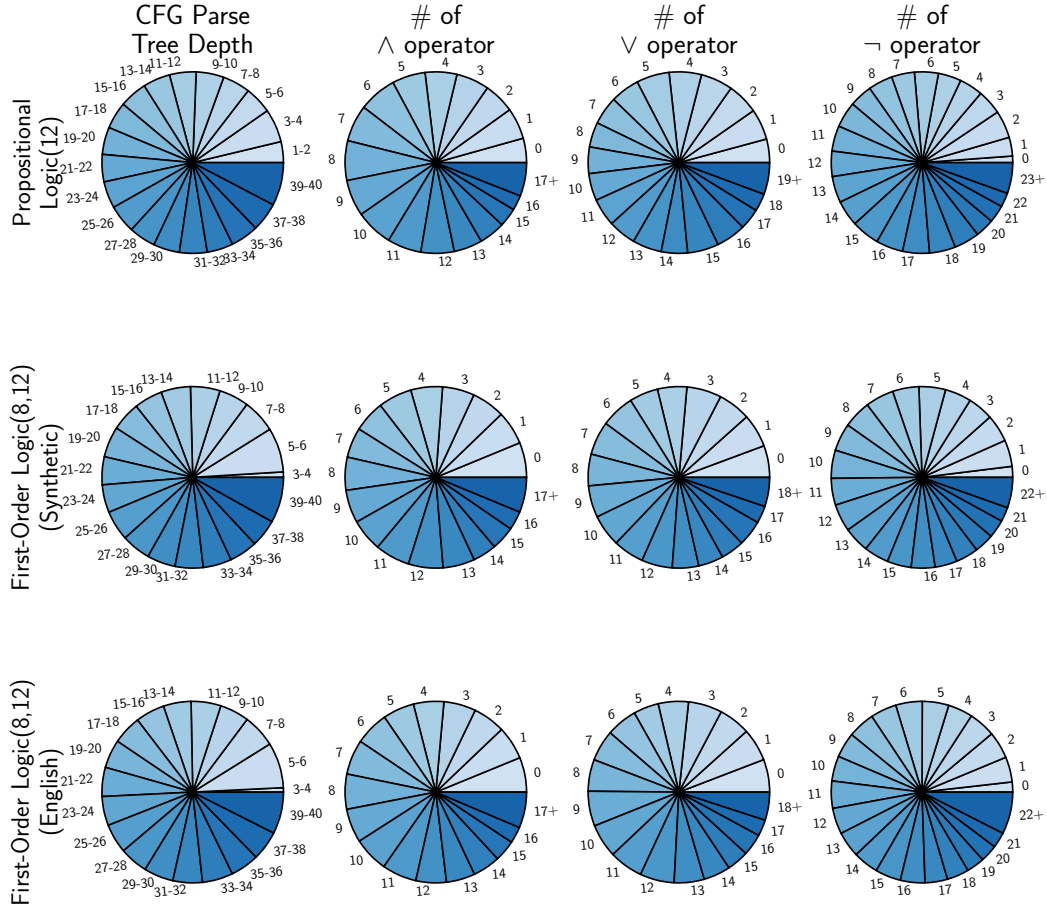


Figure 10: Count of the number of examples for each metric value for each of the datasets. Each row is a dataset and each column is a different metric that can be used to categorize the dataset. The pie charts increase in value counter-clockwise while going from lighter to darker.

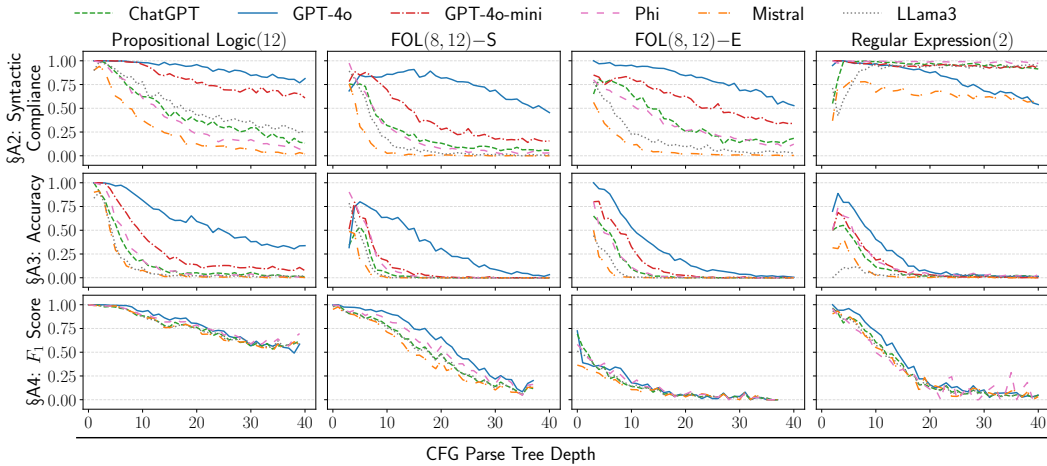


Figure 11: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using  $\forall$ uto $\exists$ \forall\mathbb{L} to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the depth of the CFG tree to produce the formula.

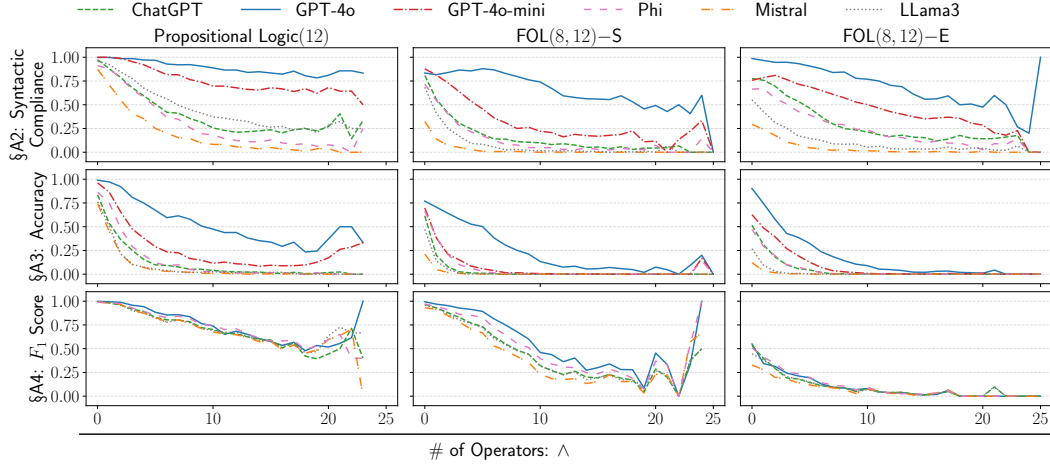


Figure 12: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using  $\forall$  to  $\exists \forall \wedge$  to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the number of and operators ( $\wedge$ ) in the expression.

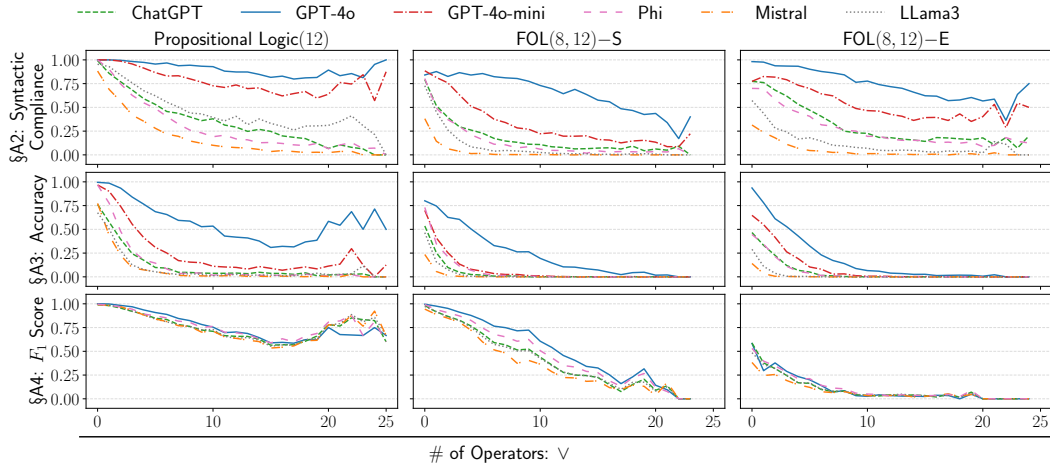


Figure 13: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using  $\forall$  to  $\exists \forall \vee$  to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the number of or operators ( $\vee$ ) in the expression.

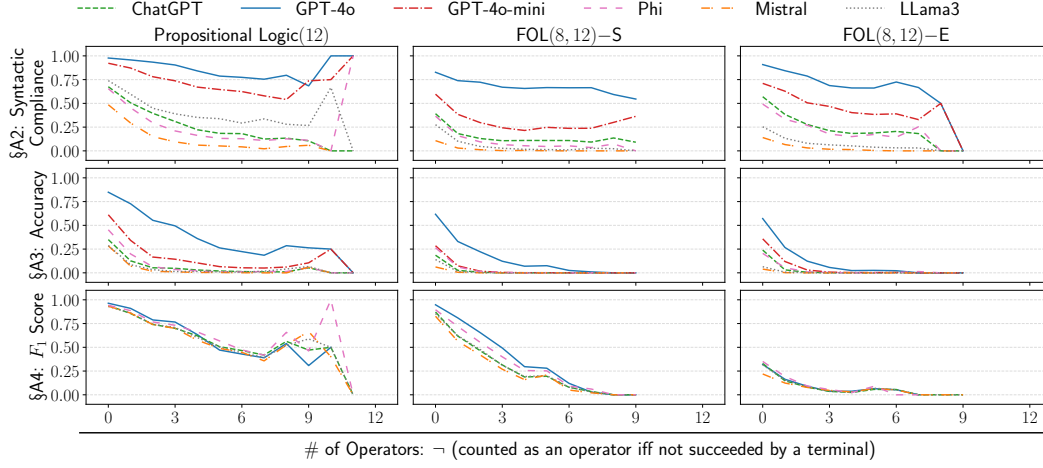


Figure 14: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using  $\forall\text{uto}\exists\text{v}\text{al}$  to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the number of negation operators ( $\neg$ ) in the expression.

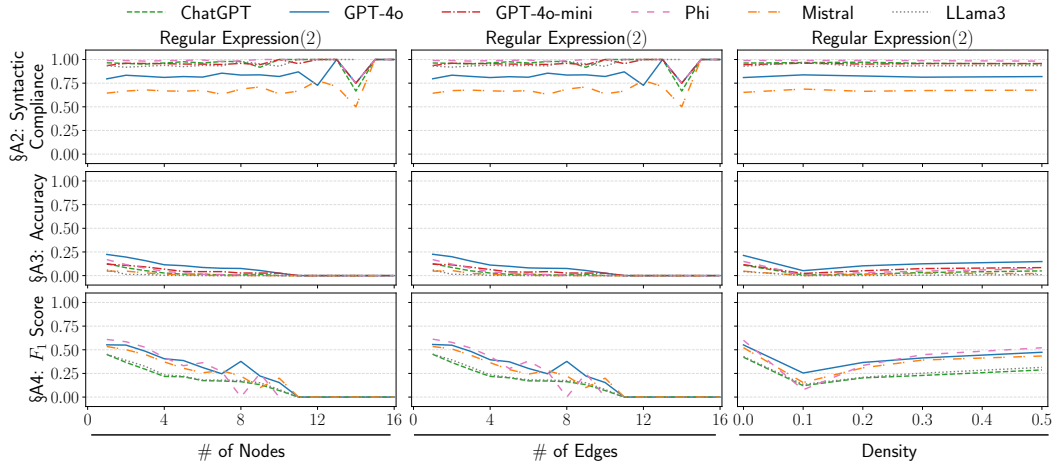


Figure 15: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using  $\forall\text{uto}\exists\text{v}\text{al}$  to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the metric on the CFG tree to produce the regular expression formula.

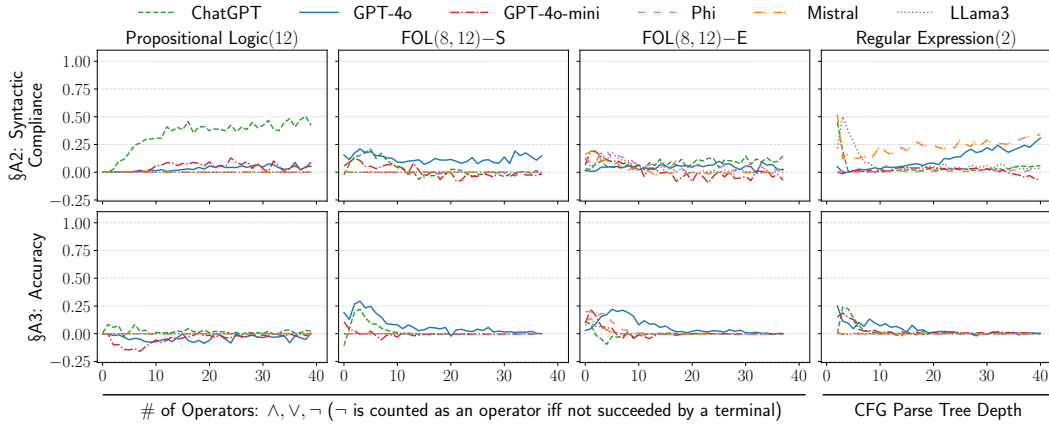


Figure 16: Syntactic compliance and accuracy difference of few-shot Pass@1 compared to zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using  $\forall\text{uto}\exists\forall\text{L}$  to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis represents the increasing order of descriptiveness complexity.

## J FEW-SHOT PROMPTING RESULTS

In this section, we discuss our few-shot prompting experiment and analyze the performance difference between zero-shot and few-shot prompting on §A1 and §A2.

We evaluated on the same five datasets from the main paper’s experiments but inserted two examples into the prompts. First-order and predicate logic used the same two examples, while regular expressions used their own two examples. In Figure 16, the performance difference of each LLM when using few-shot prompting instead of zero-shot is shown. Using few-shot prompting increases syntactic compliance as the model has access to the desired format for encoding and decoding. For expressions with lower complexity, this translates to a better performance on §A2. However, as complexity increases, the performance difference between zero-shot and few-shot prompting is negligible due to having the correct format for parsing but failing maintaining the same formula.

## K OTHER BENCHMARK CORRELATION AND $\forall\text{uto}\exists\forall\text{L}$ PREDICTIVE POWER EVALUATION

For evaluating the correlation between a LLM’s performance on  $\forall\text{uto}\exists\forall\text{L}$  and existing benchmarks and measuring the predictive power of  $\forall\text{uto}\exists\forall\text{L}$ , in Section 5, we evaluated on FOLIO (Han et al., 2022), Multi-LogicEval (Patel et al., 2024), and HumanEval (Chen et al., 2021). In this section we discuss these experiments and cite the sources of the HumanEval results along with evaluate the predictive power of  $\forall\text{uto}\exists\forall\text{L}$ .

In this section, we discuss the experimental setup for the benchmark, the sources used for LLM performance on other benchmarks, and the  $\forall\text{uto}\exists\forall\text{L}$  we used for evaluation. We also evaluate the FOLIO premise benchmark further based on the operator numbers in each premise.

### K.1 FOLIO EXPERIMENTAL SETUPS

The FOLIO dataset is composed of premises and a conclusion for each sample where the task is to conclude whether the conclusion is true, false, or unknown given the premises. Additionally, the dataset provides an encoding into first-order logic for all the premises and conclusions. Therefore, we evaluated each LLM on their abilities to (1) informalize a first-order logic premise, (2) autoformalize a natural language premise, (3) correctly classifying the conclusion using the first-order logic representations, and (4) correctly classifying the conclusion using the natural language representations.

## Prompt 9: FOLIO Premise Informalization Prompt

## [TASK]

Your task is to convert a first-order logic formula, appearing after [FORMULA], to a natural description that represents the formula. Only natural language terms are allowed to be used and do not copy the formula in your description. Your description should allow one to reconstruct the formula without having access to it, so make sure to use the correct names in your description. Explicitly describe the predicates. You may use terms verbatim as specified in the vocabulary below.

## [EXAMPLE 1]

$\forall x(\text{DrinkRegularly}(x, \text{coffee}) \vee (\neg \text{WantToBeAddictedTo}(x, \text{caffeine})))$   
 People regularly drink coffee, or they don't want to be addicted to caffeine, or both.

## [VOCABULARY]

$\vee$  represents disjunction

$\wedge$  represents conjunction

$\neg$  represents negation

$\rightarrow$  represents implication

( and ) represent parentheses

propositions can be used verbatim

predicates can be used verbatim

$\forall < x_1 > < x_2 > \dots < x_n > .$  represents universal quantification with  $x_1 \dots$  representing free variables

$\exists < x_1 > < x_2 > \dots < x_n > .$  represents existential quantification with  $x_1 \dots$  representing free variables

The objects are: **caffeine**

The parameterized predicates are: *awarethatdrug(?p0, ?p1), wanttobeaddictedto(?p0, ?p1)*

The free variables are:  $x$

## [FORMULA]

$\forall x. (\neg \text{wanttobeaddictedto}(x, \text{caffeine}) \rightarrow \neg \text{awarethatdrug}(x, \text{caffeine}))$

For the FOLIO premise informalization and autoformalization experiments, the LLM was prompted using the same few-shot first-order logic prompt used by  $\forall\text{uto}\exists\forall\wedge\text{L}$  where the example from the prompt is another premise from the same FOLIO example to make sure both the example and the evaluated premises have the same context. Premises were screened to make sure that we were able to parse them into Prover9. Below is an **example premises come from the FOLIO dataset**.

For evaluating the performance of each LLM on classifying whether the premises entailed the conclusion, the same prompt was used for both the natural language and first-order logic representations of the premises and conclusions. The prompts are inspired by the prompts used in Multi-LogiEval and use Chain-of-Thought prompting and prompt the model to provide the answer in a parsable format. An example for both premises **using an example from the FOLIO dataset are shown below**.

We evaluated the informalization results against the ground truth natural language representation using BLEU (Callison-Burch et al., 2006), ROUGE (Lin, 2004), METEOR (Banerjee & Lavie, 2005), and BERT Score (Zhang\* et al., 2020). The model deberta-xlarge-mnli (He et al., 2021) was used for the BERT score calculation. For the autoformalization results, we used the same verification process as the main paper. For the FOLIO conclusion classification, the LLM's answer was parsed out of its response with the examples that could not be parsed being classified as "Unknown" and marked as wrong. These examples were checked to verify the parser.

## K.2 MULTI-LOGIEVAL EXPERIMENT SETUP

The task in Multi-LogiEval (Patel et al., 2024) is to answer a yes-or-no question using the provided context, where the question was created using a certain depth of rules of logical reasoning. We used a prompt similar to the one they used where we use Chain-of-Thought prompting and prompt the LLM



## Prompt 10: FOLIO Premise Autoformalization Prompt

[VOCABULARY]

Use  $\vee$  to represent disjunctionUse  $\wedge$  to represent conjunctionUse  $\neg$  to represent negation

Use ( and ) to represent parentheses

The objects are: *caffeine*The parameterized predicates are: *awarethatdrug(?p0, ?p1)*,*wanttobeaddictedto(?p0, ?p1)*The free variables are: *x*

[TASK]

Your task is to interpret the natural language (NL) description of a first-order logic formula and represent it as formal syntax using the vocabulary specified in the [VOCABULARY] block above. Only output the formula and no other text. The NL description appears immediately following the [NL DESCRIPTION] tag.

[EXAMPLE 1]

*People regularly drink coffee, or they don't want to be addicted to caffeine, or both.* *$\forall x(DrinkRegularly(x, coffee) \vee (\neg WantToBeAddictedTo(x, caffeine)))$* 

[NL DESCRIPTION]

*No one who doesn't want to be addicted to caffeine is unaware that caffeine is a drug.*

## Prompt 11: FOLIO Natural Language Representation Prompt

For the following [PREMISES] containing rules of logical reasoning, perform step-by-step reasoning to answer whether the [CONCLUSION] is True/False/Uncertain based on the [PREMISES]. Use the following answer format:

Reasoning Steps:

Answer: True/False/Uncertain

[PREMISES]:

*All people who regularly drink coffee are dependent on caffeine**People regularly drink coffee, or they don't want to be addicted to caffeine, or both.**No one who doesn't want to be addicted to caffeine is unaware that caffeine is a drug.**Rina is either a student who is unaware that caffeine is a drug, or she is not a student and is she aware that caffeine is a drug.**Rina is either a student who depend on caffeine, or she is not a student and not dependent on caffeine.*

[CONCLUSION]:

*Rina doesn't want to be addicted to caffeine or is unaware that caffeine is a drug.*

to provide the answer in a specific location to parse. Examples of these prompts are provided below using examples from the Multi-LogiEval dataset.

## K.3 HUMAN EVAL AND BIG BENCH HARD SCORE SOURCES

To evaluate the correlation and predictive power of  $\forall\text{to}\exists\forall/\wedge$  against commonly used LLM benchmarks HumanEval (Chen et al., 2021) and Big Bench Hard (BBH) (Suzgun et al., 2023), we collected the performance scores of the LLMs we evaluated on both benchmarks and report our findings and sources in Table 6. We were unable to find any sources that evaluated GPT-4o-mini on BBH.



#### Prompt 12: FOLIO First-Order Logic Representation Prompt

For the following [PREMISES] containing rules of logical reasoning, perform step-by-step reasoning to answer whether the [CONCLUSION] is True/False/Uncertain based on the [PREMISES]. Use the following answer format:

Reasoning Steps:

Answer: True/False/Uncertain

[PREMISES]:

$\forall x(DrinkRegularly(x, coffee) \rightarrow IsDependentOn(x, caffeine))$   
 $\forall x(DrinkRegularly(x, coffee) \vee (\neg WantToBeAddictedTo(x, caffeine)))$   
 $\forall x(\neg WantToBeAddictedTo(x, caffeine) \rightarrow \neg AwareThatDrug(x, caffeine))$   
 $\neg(Student(rina) \oplus \neg AwareThatDrug(rina, caffeine))$   
 $\neg(IsDependentOn(rina, caffeine) \oplus Student(rina))$   
 [CONCLUSION]:  
 $\neg WantToBeAddictedTo(rina, caffeine) \vee (\neg AwareThatDrug(rina, caffeine))$

#### Prompt 13: Multi-LogicEval Prompt

"Given the context that contains rules of logical reasoning in natural language and question, perform step-by-step reasoning to answer the question. Based on context and reasoning steps, answer the question ONLY in 'yes' or 'no.' Please use the below format:

Context: *At a university, students who study hard earn high grades. Those who participate in extracurriculars develop leadership skills. However, students have restricted time outside of classes. They can either study hard or they do not develop leadership skills from extracurriculars.*

Question: *Can we conclude that Priya, a university student with limited free time, either earns high grades or does not participate in extracurricular activities?*

Reasoning steps: [generate step-by-step reasoning]

Answer: Yes/No"

## K.4 COMPUTED $\forall$ UTO $\exists$ VAL CONDITIONAL PERFORMANCE

To compare against the performance on different benchmarks in Section 5, we needed to calculate the conditional performance of each LLM's on  $\forall$ uto $\exists$ VAL for the relevant portions of the datasets. For example, there are few premises in the FOLIO dataset with more than 6 operators meaning that the most accurate comparison would be to evaluate our first-order logic dataset up to the same number of operators. Therefore we calculated the accuracy on the first-order logic formulae with less than seven operators when calculation the correlation and predictive power. On MultiLogiEval, the number of operators are dictated by the depth of the rules so we took the average of all first-order logic examples up to 30 in our dataset. On HumanEval, to the best of our knowledge using the average of regex with CFG tree depth up to 7 is the best comparison.

## K.5 FOLIO ADDITIONAL CORRELATION FIGURES

In Section 5, we evaluated the correlation of other benchmarks compared to  $\forall$ uto $\exists$ VAL. For the FOLIO dataset, we were able to calculate the exact number of operators in each problem allowing us to plot points comparing the autoformalization and informalization accuracy for each operator number class to directly compared to the accuracy of the same number operators in the first-order logic dataset we generated.

We plot these results in Figure 17 with the Pearson co-relation coefficient. Each figure shows a moderate to strong correlation with a statistically significant p-value of less than 0.05. As the computational complexity increases, performance on  $\forall$ uto $\exists$ VAL, autoformalization, and informalization decreases. The autoformalization correlation is significantly stronger due to the informalization evaluation metrics being much weaker at evaluating truth maintenance.

Table 6: Reported performance of SOTA LLMs on HumanEval and Big Bench Hard benchmarks.

Model	HumanEval Score	Big Bench Hard (BBH) Score
GPT-3.5-turbo	68 (OpenAI, 2024)	48.1 (OpenAI, 2023b)
GPT-4o	90.2 (OpenAI, 2024)	83.1 (Dunham & Syahputra, 2024)
GPT-4o-mini	87.2 (OpenAI, 2024)	-
Llama3-8B-Instruct	61.6 (Liu et al., 2023b)	24.5 (Fourrier et al., 2024)
Mistral-7b-Instruct-v0.1	42.1 (Liu et al., 2023b)	23.95 (Fourrier et al., 2024)
Phi-3-medium-4k-instruct	62.2 (Microsoft, 2024)	49.38 (Fourrier et al., 2024)

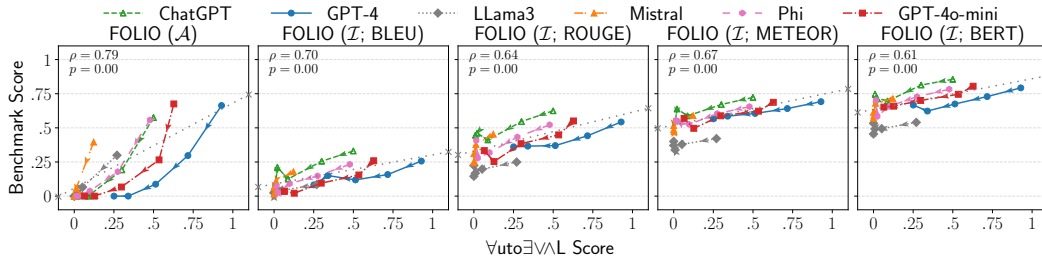


Figure 17: Correlation between scores on VutoV\A\L and both autoformalization  $\mathcal{A}$  and informalization  $\mathcal{I}$  for FOLIO premises. Each point represents a specific number of operators with arrows showing increasing complexity (number of operators). The trendline across all the points is annotated with  $\times$ , the Pearson co-relation coefficient ( $\rho$ ), and the p-value are annotated in the top left.