∀UTO∃∨∧L: AUTONOMOUS EVALUATION OF LLMS FOR TRUTH MAINTENANCE AND REASONING TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper presents ∀uto∃∨∧L, a novel benchmark for scaling Large Language Model (LLM) assessment in formal tasks with clear notions of correctness, such as truth maintenance in translation and logical reasoning. ∀uto∃∨∧L is the first benchmarking paradigm that offers several key advantages necessary for scaling objective evaluation of LLMs without human labeling: (a) ability to evaluate LLMs of increasing sophistication by auto-generating tasks at different levels of difficulty; (b) auto-generation of ground truth that eliminates dependence on expensive and time-consuming human annotation; (c) the use of automatically generated, randomized datasets that mitigate the ability of successive LLMs to overfit to static datasets used in many contemporary benchmarks. Empirical analysis shows that an LLM's performance on ∀uto∃∨∧L is highly indicative of its performance on a diverse array of other benchmarks focusing on translation and reasoning tasks, making it a valuable autonomous evaluation paradigm in settings where hand-curated datasets can be hard to obtain and/or update.

1 INTRODUCTION

Foundation Models such as Large Language Models (LLMs) have been demonstrated to successfully perform many natural language tasks involving formal syntax such as *autoformalization* – utilizing LLMs in converting natural language (*NL*) to formal syntax (*FS*) such as source code, math etc., (Wu et al., 2022; Liang et al., 2023; Guan et al., 2023), *informalization* – using LLMs to convert *FS* to *NL* (e.g. code summarization), *reasoning* – using LLMs to perform sound reasoning or derive proofs. Although these methods have been successful in small-scale scenarios, their effectiveness in maintaining truth across *NL* and *FS* remains uncertain due to the difficulty in assessing truth maintenance in such tasks. Multiple authors have noted that existing benchmarks and evaluation methodologies for such tasks are susceptible to the Benchmark Contamination Problem due to their use of static datasets, e.g, HumanEval (Chen et al., 2021; Wu et al., 2022; Han et al., 2022). One effective method to mitigate this problem in existing benchmarks is creating new data (Xu et al., 2024). However, scaling such datasets as LLMs evolve is a tedious and expensive process since their data-generation task requires expert annotators to hand-generate well-balanced datasets. Moreover, such benchmarks often rely on insufficient/incomplete measures of evaluation (e.g, BLEU scores (Callison-Burch et al., 2006), ranking disparities in LLM-generated code on test cases in HumanEval vs HumanEval+ (Liu et al., 2023a)), and thus, provide misleading signals on LLM capabilities.

This paper addresses three key desiderata for benchmarking LLM capabilities for truth maintenance across *NL* and *FS*: *(D1) Can we dynamically generate out-of-distribution datasets without relying on human annotators? (D2) How do we accurately assess an LLM's truth maintenance capabilities? (D3) Can our metric serve as a predictor of LLM performance in FS-based tasks?*

For §D1, we introduce a new approach that utilizes context-free grammars to generate well-balanced, out-of-distribution datasets on the fly. For §D2, we perform closed-loop testing of LLM capabilities using formal verifiers to automatically evaluate its truth maintenance capabilities. To answer §D3, we show that our metrics can serve as predictors of LLM performance on other, well-known benchmarks.

Main contributions Our key contributions are as follows:

1. A new, dynamic approach for automatic synthesis of well-balanced test datasets that are unlikely to be memorized or seen during the LLM's training process.

2. The utilization of formal verifiers such as theorem provers to provably validate syntax-independent notions of correctness without having to exhaustively test over all possible truth valuations of formal syntax involving logic.
3. ∀uto∃∨∧L: a scalable, plug-and-play assessment system for benchmarking new LLMs as and when they are developed. Our system can be extended to any class of formal syntax that uses a grammar and admits an equivalence checker.
4. We show that LLM performance on our metric serves as an effective indicator of LLM performance on other metrics across a wide variety of tasks such as first-order logic reasoning, etc. Thus, our metric offers a scalable and efficient surrogate for evaluating new LLMs in such tasks where other metrics may be limited due to the unavailability of new datasets. Our empirical evaluation shows that SOTA LLMs are unable to maintain truth effectively.

2 FORMAL FRAMEWORK

(Large) Language Models (L)LMs LMs are non-linear functions represented by (billons of) parameters θ that, given a set of input tokens $x_1, \ldots, x_n$, typically representing *NL*, predict the output token y_{i+1} using the distribution $P(y_{i+1}|x_1, \ldots, x_n, y_1, \ldots, y_i; \theta)$. The input tokens contains *context* κ (also known as a prompt) that provides the necessary information for the task (e.g., instructions, etc). It is known that κ significantly impacts the response quality $y_1, \ldots, y_n$ (Sahoo et al., 2024).

Propositional Logic Propositional logic is a branch of logic that utilizes *propositions* and *logical operators* (e.g., conjunction: $\wedge$, etc) to construct sentences that can be used to perform reasoning using the rules of logic. For example, propositions, $p_1 =$ *It is raining*, $p_2 =$ *It is sunny* can be used to create a sentence $P = p_1 \vee p_2$. If P is true and $\neg p1$ is observed, then one can use the rules of inference to deduce that p_2 is true (Huth & Ryan, 2004).

Equivalence in Propositional Logic Two sentences in propositional logic, P_1 and P_2, are equivalent, $P_1 \equiv P_2$, iff their truth values agree for all possible assignments. E.g., $\neg(p_1 \wedge p_2) \equiv \neg p_1 \vee \neg p_2$ since $\forall p_1, p_2 \in \{\mathit{True}, \mathit{False}\} \times \{\mathit{True}, \mathit{False}\}$, $\neg(p_1 \wedge p_2) = \neg p_1 \vee \neg p_2$.

First-order Logic (FOL) FOL differs from propositional logic in that sentences are constructed using *predicates*, *quantifiers*, and *objects*. A popular example is the syllogism where, given two first-order logic sentences $\forall x.\, \mathit{Man}(x) \rightarrow \mathit{Mortal}(x)$ and $\mathit{Man}(\mathit{Socrates})$, one can conclude that $\mathit{Mortal}(\mathit{Socrates})$. A first-order logic sentence F can be interpreted using a universe $\mathcal{U}$, a substitution operator σ, and an interpretation function $\mathcal{I}$ (Russell & Norvig, 2020).

Equivalence in First-order Logic Two sentences, F_1, F_2 in first-order logic are equivalent, $F_1 \equiv F_2$, iff they are equivalent under all possible models. E.g., $\neg\forall x.\, \mathit{Man}(x) \equiv \exists y.\, \neg \mathit{Man}(y)$.

Regular Expressions A regular expression (regex) is a sequence of characters that can be used to determine whether a particular string matches the pattern or *language* induced by the regex. For example, the regex $200(00)^*1$ using $\Sigma = \{0, 1, 2\}$ matches all strings possible using Σ that begin with a two, followed by one or more pairs of zeroes, and end with a one (Hopcroft et al., 2001).

Equivalence between Regular Expressions Two regexes, R_1 and R_2 are equivalent, $R_1 \equiv R_2$, if they represent the same language. It is known that $R_1 \equiv R_2$ if their corresponding minimal deterministic finite automata (DFAs), D_1, D_2, are isomorphic, i.e., $D_1 \simeq D_2$ (Hopcroft et al., 2001).

We refer to sentences (strings) in first-order and propositional logic (regexes) as formal syntax *FS* in this paper. We now provide a definition of *(Auto/In)formalization* in the context of LLMs and *FS*.

Definition 2.1 (Autoformalization: $\mathcal{A}$)**.** Given an LLM L, an *NL* description ψ and context κ, autoformalization, $\mathcal{A}_L(\psi, \kappa)$, is defined as using L to translate ψ to *FS* φ s.t. $\mathcal{A}_L^{-1}(\varphi, \kappa') \equiv \psi$.

Example One possible autoformalization of *"Every human drinks coffee but some are not dependent on it"* in FOL is $\forall x.\, \mathit{Human}(x) \implies \mathit{Drinks}(x, \mathit{Coffee}) \wedge \exists y.\, x = y \wedge \neg \mathit{Dependent}(y, \mathit{Coffee})$.

Definition 2.2 (Informalization: $\mathcal{I}$)**.** Given an LLM L, an expression φ in *FS* and context κ, informalization, $\mathcal{I}_L(\varphi, \kappa)$, is defined as using L to translate φ to *NL* ψ s.t. $\mathcal{I}_L^{-1}(\psi, \kappa') \equiv \varphi$.

Example It is easily seen that informalization is the inverse of autoformalization. Therefore, the FOL formula $\forall x.\, \mathit{Human}(x) \implies \mathit{Drinks}(x, \mathit{Coffee}) \wedge \exists y.\, x = y \wedge \neg \mathit{Dependent}(y, \mathit{Coffee})$ can be informalized to the sentence *"Every human drinks coffee but some are not dependent on it"*.

Note that we only require $\mathcal{A}^{-1}(\varphi, \kappa) \equiv \psi$ and $\mathcal{I}_L^{-1}(\psi, \kappa') \equiv \varphi$. It is possible that a different LLM (or the same with a different seed) would autoformalize the same example to a syntactically different but semantically equivalent formula – $\forall x.\, \mathit{Human}(x) \implies \mathit{Drinks}(x, \mathit{Coffee}) \land \neg\forall y.\, \mathit{Human}(y) \implies \mathit{Dependent}(y, \mathit{Coffee})$. Similarly, an LLM can informalize differently. The example above could be informalized by the same LLM to *"**All humans drink coffee** but some are not dependent on it"*. Thus, it is not necessary that $\mathcal{A}_L(\varphi, \kappa) = \mathcal{I}_L(\psi, \kappa')$ and vice versa. We assume that the context κ, κ' provided contains the prompt and any necessary vocabulary that is needed for the task (e.g., $\mathit{Human}(x)$ represents that x is a human, etc.). Henceforth, unless specified, we skip κ, κ' and L in the notation for $\mathcal{A}$ and $\mathcal{I}$.

Given an LLM L, we define truth maintenance as L's ability to be able to understand its own translations. Given $n \in \mathbb{N}^+$, we use $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$, to refer to a sequence $\varphi_0 \rightarrow \psi_0 \rightarrow \ldots \rightarrow \varphi_n$ that is obtained using L when starting with *FS* φ_0, where $\psi_i = \mathcal{I}(\varphi_i)$ and $\varphi_{i+1} = \mathcal{A}(\psi_i)$.

Definition 2.3 (LLM Truth Maintenance w.r.t. $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$)**.** Given an LLM L and a sequence $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$ obtained using L, we define truth maintenance as $\varphi_i \equiv \varphi_j$ for any $i, j \in \{0, \ldots, n\}$.

The Need for Truth Maintenance The ability of an LLM to maintain truth across $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$ for *FS* such as first-order logic, etc., is foundational and underlies many aspects of the capabilities of LLMs surrounding reasoning, semantically accurate translation, etc. In fact, for programming, it has been shown that autoformalization can help with the reasoning abilities of LLMs since they frame reasoning as generation of *FS* (Chen et al., 2021). Others (Wu et al., 2022) have made similar observations and have highlighted the need for benchmarks and metrics for assessing the truth maintenance capabilities of LLMs. In this paper, we further show through our empirical evaluation that truth maintenance on these types of *FS* is indicative of performance on related tasks.

Naturally, LLMs may not autoformalize, reason, etc., correctly due to issues such as hallucination (Ji et al., 2023), etc. For the example earlier, the LLM could autoformalize by omitting the $x = y$ statement to yield $\forall x.\, \mathit{Human}(x) \implies \mathit{Drinks}(x, \mathit{Coffee}) \land \exists y. \neg \mathit{Dependent}(y, \mathit{Coffee})$. This seems innocuous but changes the meaning since y is no longer required to be a human, and thus it interprets as *"All humans drink coffee, and, there are some elements of the universe that are not dependent on coffee."* Such issues have profound implications in synthesizing specifications and/or programs. Thus, an LLM must be able to understand its own generated output across *NL* and *FS*, and it is imperative to create a benchmark that can faithfully assess the truth maintenance of LLMs w.r.t. $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$.

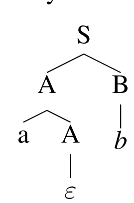
3 Our Approach for Assessing Truth Maintenance

We now describe our approach, ∀uto∃∨∧L, for autonomously assessing an LLM's ability to maintain truth w.r.t. $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$. ∀uto∃∨∧L provides dynamically generated datasets that can be scaled arbitrarily by systematically generating out-of-distribution, well-balanced ground-truth data (§D1 – Sec. 1), provides §D2 by using intrinsic LLM capabilities to automatically assess $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$ without requiring any labeled annotations and using formal verifiers to rigorously check and guarantee the correctness of $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$ without having to engage in an exhaustive search process.

Dynamic Dataset Generation We use context-free grammars (CFGs) (Hopcroft et al., 2001) – a set of *production rules* over *terminal* and *non-terminal* symbols – for dynamically generating datasets.

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow aA|\varepsilon \\
B &\rightarrow b
\end{aligned}
$$

(*a*) CFG $\mathcal{G}$ for the language a^*b



(*b*) Parse tree when using $\mathcal{G}$ to obtain the string ab ($d = 2$).

An example CFG $\mathcal{G}$ with 3 production rules and the *infix parse tree* that is obtained by repeatedly applying the rules to yield the string ab is illustrated above. The depth of this tree is often used to measure the *descriptional complexity* d of a given string generated using the CFG. (Csuhaj-Varjú & Kelemenová, 1993). CFGs can be used to dynamically generate arbitrarily large amounts of data.

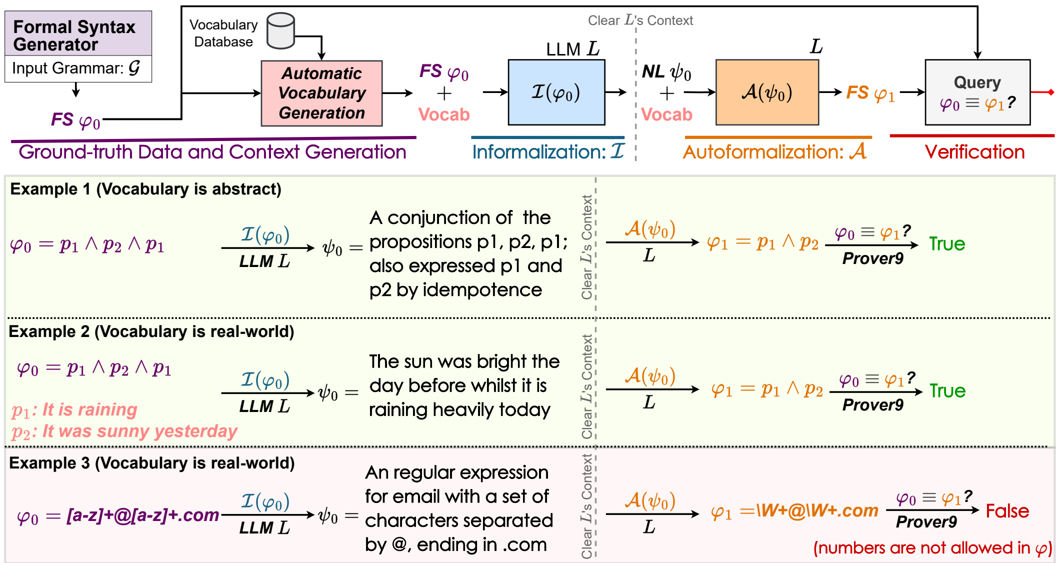


Figure 1: The ∀uto∃∨∧L pipeline for autonomous evaluation of LLM truth maintenance w.r.t. $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$.

Another advantage is that CFGs can be customized with minimal human effort to generate diverse datasets whose ground-truth data possesses specific properties. For example, a dynamic dataset that only consists of k–SAT sentences – propositional logic in the Canonical Normal Form $(P_1^0 \vee \ldots \vee P_k^0) \wedge (P_1^1 \vee \ldots \vee P_k^1) \wedge \ldots$ where $P_i^j \in \{p_x, \neg p_x\}$ – can be easily generated. We enrich the generated sentence with context via a customizable *Vocabulary Generation* step, which automatically provides the necessary vocabulary for performing the task (e.g., providing English meanings to allow for human-like *NL*) by using terms from a vocabulary database or by using an LLM.

Automatic Truth Maintenance w.r.t. $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$ We develop a novel technique that can soundly assess truth maintenance without any human annotations by evaluating $\varphi_i \rightarrow \psi_i \rightarrow \varphi_{i+1}$. Our approach is based on the following intuition. Let $\mathcal{I} : \varphi \rightarrow \psi$ be a non-deterministic function that maps *FS* φ to *NL* ψ. Similarly, let $\mathcal{A} : \psi \rightarrow \varphi$ be a non-deterministic function that maps *NL* ψ to *FS* φ. In general, there are many possible correct informalizations (autoformalizations) of $\varphi \in$ *FS* ($\psi \in$ *NL*). Thus, $\mathcal{I}$ and $\mathcal{A}$ are not injective (1:1) functions and thus $\mathcal{I}^{-1}$ and $\mathcal{A}^{-1}$ are not well-defined.

Our key observation is that if $\mathcal{I}$ and $\mathcal{A}$ come from the same system (e.g., an LLM), then we can evaluate its truth maintenance by *composing* $\mathcal{I}$ and $\mathcal{A}$. Let φ be any *FS* expression and let L be an LLM. Now, if L preserves truth, then $\psi = \mathcal{I}(\varphi)$ will be an accurate *NL* representation of φ and $\varphi' = \mathcal{A}(\psi)$ will be a semantically equivalent *FS* representation of ψ. Since ψ is an *NL* description, it is quite challenging to check whether $\mathcal{I}(\varphi)$ is indeed an accurate representation of φ without human intervention. However, if L preserves truth, $\varphi' = \mathcal{A}(\mathcal{I}(\varphi))$ will be semantically equivalent to φ even if they are not syntactically identical. Thus, we only need to check if $\varphi \equiv \varphi'$. For example, let $\varphi_0 = p_1 \wedge p_1$, $\psi_0 = \mathcal{I}(\varphi_1) =$ *"A conjunction of propositions p_1 and p_1 that can be simplified to p_1 using Idempotence."*, and $\varphi_1' = \mathcal{A}(\psi_0) = p_1$ for a sequence $(\mathcal{A} \circ \mathcal{I})^1(\varphi_0)$. It is very difficult to check if ψ_0 is an accurate representation of φ_0, but easy to check if $\varphi_0 \equiv \varphi_1$ using a formal verifier.

Formal Verification Since ∀uto∃∨∧L uses formal syntax φ as input and produces formal syntax φ' as output, we can use formal verifiers to check whether $\varphi \equiv \varphi'$. As a result, ∀uto∃∨∧L avoids brittle syntactic equivalence checks and exhaustive tests of semantic equivalence that require evaluations of all possible truth valuations of formulas or executions of regexes.

∀uto∃∨∧L Overall Pipeline We use the above insights to automatically assess LLM truth maintenance by using the same LLM L to represent $\mathcal{I}$ and $\mathcal{A}$ respectively. Fig. 1 shows our overall assessment process. Briefly, we use a CFG $\mathcal{G}$ to automatically generate a ground-truth *FS* expression φ_0. Next, we use a vocabulary generation process to generate a context for φ_0. This can either use abstract terms or use *NL* elements for more human-like scenarios (§D1). We then evaluate $(\mathcal{A} \circ \mathcal{I})^1(\varphi_0)$ by using an LLM L to first generate $\psi_0 = \mathcal{I}(\varphi_0, \kappa)$ using context κ designed for informalization. The context of L is cleared (note that we only use the output of $\mathcal{I}(\varphi_0)$), and we use

(a) 3–SAT	(b) Propositional Logic	(c) First-order Logic	(d) Regular Expression
$S \to S \wedge S$	$S \to (S \wedge S) \mid (S \vee S)$	$S \to F \mid (\forall f.\, S) \mid (\exists f.\, S)$	$S \to (S)K \mid \Sigma K$
$S \to (P \vee P \vee P)$	$S \to (\neg S)$	$F \to (F \wedge F) \mid (F \vee F)$	$S \to S\Sigma K$
$P \to \neg v \mid v$	$S \to \neg v \mid v$	$F \to (\neg F) \mid \neg p \mid p$	$K \to * \mid \varepsilon$

Figure 2: CFGs (described in Sec. 4) used for synthesizing the datasets in ∀uto∃∨∧L.

L to generate $\varphi_1 = \mathcal{A}(\psi_0, \kappa')$ using context κ' designed for autoformalization. We then use a verifier (e.g., Z3 (de Moura & Bjørner, 2008), Prover9 (McCune, 2010)) to assess if $\varphi_0 \equiv \varphi_1$ since both are elements of *FS*. If $\varphi_0 \equiv \varphi_1$ then we can repeat the process by evaluating $(\mathcal{A} \circ \mathcal{I})^1(\varphi_1)$ similarly.

Example: Consider example 2 in Fig. 1. ∀uto∃∨∧L uses the grammar in Fig. 2b to automatically generate a ground truth *FS* sentence as $\varphi_0 = p_1 \wedge p_2 \wedge p_1$. We can use any vocabulary to generate meaning for the propositions; p_1 : *It is raining today*, p_2 : *It was sunny yesterday*. Next, the LLM L is prompted with Prompt 1 to perform informalization yielding *NL* $\psi_0 = \mathcal{A}(\varphi_0)$. L can perform any simplification or other paraphrasing necessary. For example, L could informalize φ_0 above to ψ_0 =*"The weather status was sunny yesterday whilst it is raining today."* Notice that the LLM-generated *NL* statement automatically reflects a simplification using the Commutative $(a \wedge b \equiv b \wedge a)$ and Idempotent $(a \wedge a \equiv a)$ properties. Next, L is asked to autoformalize ψ_0 without any context other than the vocabulary to use and a prompt for autoformalization (Appendix F). In this case, the LLM could return $\varphi_1 = \mathcal{A}(\psi_0) = p_1 \wedge p_2$. We use a theorem prover such as Prover9 (McCune, 2010) to show that $\varphi_0 \equiv \varphi_1$ and thus assess L's truth maintenance capabilities w.r.t. $(\mathcal{A} \circ \mathcal{I})^1(\varphi_0)$.

4 Datasets and Assessment Framework

∀uto∃∨∧L is open-source[1], is written in Python 3, includes several pre-computed datasets, and is easily customizable for adding new datasets, prompts, LLMs, etc. We now describe the datasets and metrics that any newly developed LLM can be evaluated on by using ∀uto∃∨∧L out-of-the-box.

Pre-generated Datasets and Dynamic Dataset Generator We provide 5 datasets using the grammars in Fig. 2. All datasets are arranged based on the descriptional complexity d (# of operators for logic, parse tree depth for regex) with around 500 samples per complexity for a total of $20k$ samples per dataset. Other dimensions for categorization are available as metadata.

(Fig. 2a) k–SAT(n) Dataset ($|\mathcal{D}^*| \sim 10k$) The terminal v is replaced with a vocabulary of n propositions $p_1, \ldots, p_n$. This dataset is used for prompt calibration due to its toy structure.

(Fig. 2b) Propositional Logic: PL(n) Dataset ($|\mathcal{D}^*| \sim 19k$) Similar to k–SAT, this dataset also replaces terminals by randomly selecting from a list n propositions.

(Fig. 2c) First-order Logic: FOL(n_p, n_o) Synthetic (S), English (E) Datasets ($|\mathcal{D}^*| \sim 19k$ each) The terminals p are replaced with predicates of the form $p(v_1, \ldots, v_n)$ where p_i is a predicate name selected from a list of n_p predicates, v_i is either an object o from a list of n_o objects or is a free variable $f \in \{x_1, x_2, \ldots\}$ that is appropriately annotated within the scoping rules. The objects and predicate names are auto-generated synthetically for the synthetic version of the dataset. The English version of the dataset uses VerbNet (Schuler, 2005) for predicate names and Faker (Faraglia, 2024) for object names. The English dataset allows for informalization to produce more *abstract* sentences that closely resemble the *NL* statements in SOTA autoformalization datasets. For example, an *FS* statement *Boom(Richard)* ∧ *Exercise(Yolonda)* yields a more natural *NL* statement such as *"The expression states that Richard does not experience a boom, and Yolonda does not engage in exercise"*.

(Fig. 2d) Regular Expression: RE(n) Dataset ($|\mathcal{D}^*| \sim 18k$) The vocabulary Σ is the set $\{0, \ldots, n-1\}$ where n is a user-specified constant.

Dataset Diversity Our overall dataset's total number of unique samples $|\mathcal{D}^*|$ is $\sim 85k$. We also provide zero-shot and 2-shot prompts for each dataset, making the total dataset size $170k$ for off-the-shelf evaluation and continual assessment of any new LLMs. Similarly, ∼85% of the samples in all datasets are composed of unique CFG parse trees (trees obtained by sampling the CFG but not

[1]Source code (and appendix) is included in the supplement. We will make it public post acceptance.

Prompt 1: Informalization ($\mathcal{I}$) for Logic (others available in Appendix F)

Your task is to convert a ⟨Propositional Logic, 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.

[VOCABULARY]

Operators:	List of operators followed by their NL interpretations
Objects:	The objects in the universe (if any)
Propositions:	The propositions in the universe and their NL interpretations (if any)
Predicates:	The predicates in the universe and their NL interpretations (if any)
Examples:	Few-shot examples of the task (if any)

Example Prompt

Your task . . .

Operators:	$\wedge$ represents conjunction, $\vee$ represents disjunction, . . .
Propositions:	p_1 : *It is raining*, p_2 : *It was sunny yesterday*
Formula:	$p_1 \wedge p_2 \wedge p_1$

Example Response: `The sun was bright the day before whilst it is raining today.`

injecting the vocabularies). Expressions with the same parse tree but different vocabularies such as $p_1 \wedge p_2$, $p_2 \wedge p_1$, $((p_2) \wedge p_1))$, etc., compose $\sim$10% of our dataset. Such samples provide a robust check against positional bias in the LLM.

Efficient Dynamic Dataset Generation Users can easily generate datasets in ∀uto∃∨∧L by simply providing CFGs and/or vocabularies. We provide a dataset generator (described in App. B) that can use a user-provided input CFG and vocabulary to dynamically generate user-controlled, diverse datasets up to a user-specified metric such as number of operators, parse tree depth, etc. Our generator is guaranteed to be able to generate any string that is representible using the CFG.

Robust Parser We use open-source libraries to robustly parse the LLM-generated output. We use the Natural Language Toolkit (NLTK) library (Bird et al., 2009) for parsing logic and use Reg2Dfa (Reg, 2017) for regexes. LLM output that cannot be parsed is said to be *syntactically non-compliant*. Additionally, we also use scripts to ensure that the informalization step does not copy elements of *FS* into *NL* (e.g., complete or any parts of *FS*) that would otherwise make autoformalization trivial.

Prompt Calibration As stated in Sec. 2, prompts are crucial for LLM performance. To ensure that our results can be viewed in terms of LLM capabilities themselves and not a characteristic of using "bad" prompts, we conducted extensive prompt engineering and ensured that at least one LLM could perform ∀uto∃∨∧L with $\geq 95\%$ accuracy on a constrained (but representative) grammar, e.g., 3-SAT(12) (Fig. 7). For §A4, rather than asking for only a yes or no answer, we use Chain-of-Thought (CoT) so that LLMs can utilize their generated outputs to improve their reasoning (Wei et al., 2022).

Evaluation Metrics ∀uto∃∨∧L automatically assesses LLMs and provides reports that help answer:

A1. Is performance on ∀uto∃∨∧L indicative of performance on other benchmarks We compute the correlation and predictive power of ∀uto∃∨∧L w.r.t. other benchmarks.
A2. Are LLMs syntactically compliant? ∀uto∃∨∧L evaluates the ability of LLMs to generate syntactically correct output by computing the ratio of generated *FS* that could be successfully parsed.
A3. Are LLMs able to maintain truth w.r.t. $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$*?* We compute a quantitative measure of LLM truth maintenance w.r.t. $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$ by computing the accuracy of an LLM to yield *FS* φ_1 s.t. $\varphi_0 \equiv \varphi_1$ using the ∀uto∃∨∧L pipeline (we used $n = 1$ for our evaluation to keep costs low).
A4. Can LLMs be used as verifiers? ∀uto∃∨∧L evaluates whether an LLM L can be used to provide the answer to $\varphi_0 \equiv \varphi_1$ instead of using a formal verifier during $(\mathcal{A} \circ \mathcal{I})^1(\varphi_0)$.

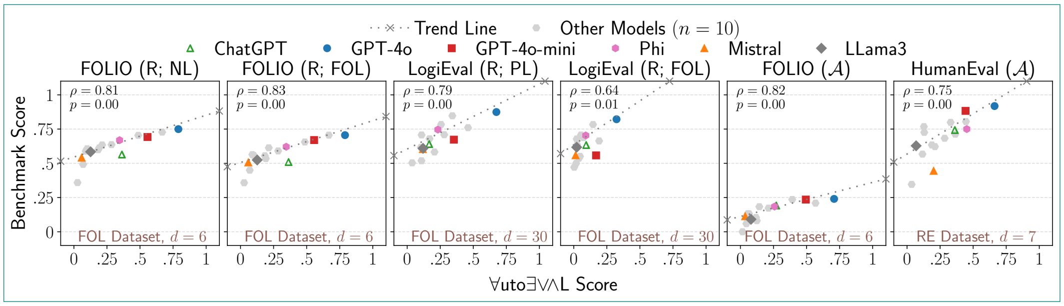


Figure 3: Correlation between scores on ∀uto∃∨∧L and static benchmarks from the literature. The Pearson correlation coefficient (ρ) and the p-value (values ≤ 0.05 are statistically significant) are annotated in the top left. ∀uto∃∨∧L scores use a comparable descriptional complexity d (App. K.4). R represents a reasoning task. Grey hexagons (⬣) represent data from 10 other models (tabular data is included in Appendix).

5 Assessment of SOTA LLMs on the ∀uto∃∨∧L Benchmark

To motivate research in the area and showcase our framework's strengths, we used ∀uto∃∨∧L to evaluate 17 SOTA closed and open-source LLMs of various parameter sizes. For clarity, we plot select models, grey out the data from the others, and refer the reader to the Appendix for a comprehensive view. We analyze §A1 using Fig.3 and Fig. 4. We analyze §A2, A3, and A4 using results obtained by using ∀uto∃∨∧L on our generated datasets (Fig. 5). We present our analyses below.

§A1: *Is performance on ∀uto∃∨∧L indicative of performance on other benchmarks* Our hypothesis is that the ability for truth maintenance on foundational concepts (propositional logic, first-order logic, regexes, etc.) will be indicative of an LLM's reasoning abilities. To evaluate this, we compare the performance of LLMs on ∀uto∃∨∧L vs. performance in other benchmarks focused on *FS*-tasks such as reasoning, autoformalization, etc. Our results (Fig. 3) indicate that there is a positive correlation between LLM performance on ∀uto∃∨∧L and other logic-based benchmarks on a myriad of tasks such as autoformalization, logical reasoning, code generation, etc.

We use 5 popular benchmarks: (a) FOLIO(R;{NL,FOL}) (Han et al., 2022), a popular logical reasoning benchmark with ground truth in both *NL* and *FS*; (b) FOLIO($\{\mathcal{A}/\mathcal{I}\}$) evaluates if an LLM can (auto/in)formalize *NL* (*FS*) accurately; (c) LogiEval(R;{PL,FOL}) (Patel et al., 2024) a reasoning benchmark with ground truth in propositional and first-order logic; (d) HumanEval($\mathcal{A}$) (Chen et al., 2021), a code autoformalization benchmark; (e) Big Bench Hard (BBH) (Suzgun et al., 2023). These benchmarks are contrasted in Sec. 6, and example prompts of these benchmarks are included in Appendix K. We ran 5 runs across all these benchmarks except BBH (due to resource limitations) using the most comparable ∀uto∃∨∧L dataset. For BBH, we use the reported numbers in the literature as scores for the models (sources are included in Appendix K).

Our results (Fig. 3) show that ∀uto∃∨∧L exhibits a strong, positive correlation $\rho \geq 0.7$ with other static benchmarks on *FS*-based tasks and even on reasoning tasks such as FOLIO. However, ∀uto∃∨∧L evaluates truth maintenance capabilities of LLMs by automatically generating its own data and without requiring any hand-annotation. Similar results using ∀uto∃∨∧L can be observed in LogiEval for propositional logic. Our results only showcase a moderate correlation ($0.5 \leq \rho < 0.7$) for the FOL version of LogiEval. We investigated and found that GPT-4o-mini was unable perform longer chains of reasoning on the FOL problems in LogiEval (LogiEval includes problems that require multiple reasoning chains with different rules of inference). Similarly, ∀uto∃∨∧L also serves as a predictor for autoformalization, as evident in our results on FOLIO ($\mathcal{A}$) and HumanEval.

Definition 5.1 (Predictive Power: $\mathcal{P}_{|Y}$)**.** Given two benchmarks X and Y, where LLMs L_1, L_2 score x_1, x_2 on X and score y_1, y_2 on Y respectively, the predictive power of Y w.r.t X, $\mathcal{P}_{|Y}(X)$, is the probability that $x_1 \geq x_2$ if $y_1 \geq y_2$. Formally, the predictive power $\mathcal{P}_{|Y}(X) = P(x_1 \geq x_2 | y_1 \geq y_2)$.

Predictive Power ∀uto∃∨∧L can be used as a performance predictor for other metrics. We evaluated this capability using the predictive power as defined above. The probabilities were obtained using Maximum Likelihood Estimation over $\langle$∀uto∃∨∧L, $M\rangle$ and $\langle M,$∀uto∃∨∧L$\rangle$ for a metric M.

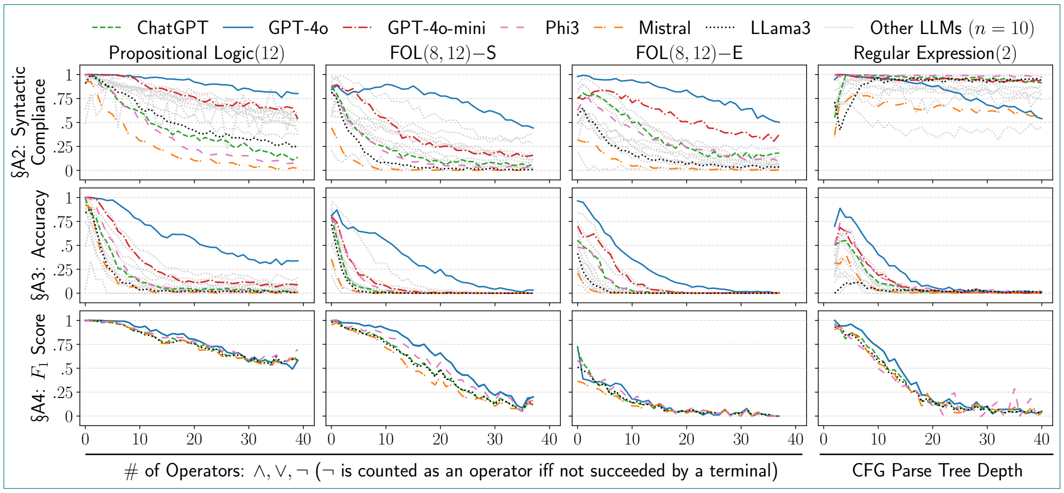


Figure 5: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using ∀uto∃∨∧L to assess LLMs w.r.t. §A2, §A3, §A4 on the packaged datasets (Sec. 4). The x-axis represents an increasing descriptional complexity. Our prompt calibration, few-shot, other model results, etc. are included in the appendix.

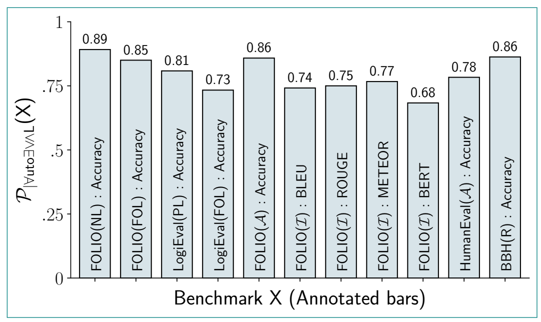


Figure 4: Predictive power of ∀uto∃∨∧L w.r.t other benchmarks. Benchmark metrics appear after the colon.

Our results (Fig. 4) show that an LLM's performance on ∀uto∃∨∧L is a strong predictor of its performance on *FS*-based benchmarks. Our metric is also more robust at measuring truth maintenance than *NL*-based metrics like BLEU scores. For example, in a FOLIO($\mathcal{I}$) informalization task, changing the generated *NL* ψ =*"the weather status was sunny yesterday and is raining today"* to ψ' =*"the weather status was sunny yesterday and is not raining today"* still achieves a high BLEU(ψ', ψ) score of 0.74 (BLEU$(\psi, \psi) = 1$) but does not maintain truth. Even for such metrics, $\mathcal{P}_{|\forall\text{uto}\exists\vee\wedge\text{L}} > 0.5$.

These results show that performance on ∀uto∃∨∧L is indicative of performance on other benchmarks.

§A2: *Are LLMs syntactically compliant?* As seen in Fig. 5, SOTA LLMs can perform $(\mathcal{A} \circ \mathcal{I})^1(\varphi_0)$ well when the formal syntax has low descriptional complexity (e.g., few operators in logic). But, as the descriptional complexity increases, the ability of LLMs to autoformalize their own informalizations decreases. One surprising result here is that for regexes, GPT-4o performs much worse than Phi and LLama-3, which are much smaller models. We observed that GPT-4o tends to expand the Kleene Star recursively, leading to invalid regexes. For logic, we observed that LLMs do not make mistakes in the operators or symbols used but often misplace parentheses, creating malformed expressions.

§A3: *Are LLMs able to maintain truth w.r.t.* $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$*?* Our results show that, except for the prompt calibration task, LLMs cannot perform even $(\mathcal{A} \circ \mathcal{I})^1(\varphi_0)$ well as the descriptional complexity increases. One common failure is the lack of understanding of precedence and associativity rules for the formal syntax. The evaluated LLMs often cannot place the correct operators in the correct scope, leading to quick verification failures. We provide an analysis of failing cases in Appendix G.

Bounding the false positive rate of ∀uto∃∨∧L One key advantage of ∀uto∃∨∧L is that it is robust against different informalizations of the same *FS*. Thus, when ∀uto∃∨∧L outputs that LLM maintains truth $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$ on *FS* φ_0, the intermediate *NL* $= \mathcal{I}(\psi_0)$ is a semantically equivalent translation of φ_0. We now bound the probability of false positives that may occur due to hallucinations.

Given an LLM L, let $\varphi_0 \xrightarrow{\mathcal{I}_L(\varphi_0)} \psi_0 \xrightarrow{\mathcal{A}_L(\psi_0)} \varphi_1$ be an execution of the ∀uto∃∨∧L pipeline for $(\mathcal{A} \circ \mathcal{I})^1(\varphi_0)$ s.t. $\varphi_0 \equiv \varphi_1$ but ψ_0 is not an accurate representation of φ_0. We statistically analyze the chance of ∀uto∃∨∧L providing such false positives. Let $p_{\mathcal{I}}$ be the probability with which L informalizes an *FS* expression $\mathcal{I}(\varphi_0) = \psi_0$ s.t. ψ_0 is an accurate representation of φ_0. Similarly, let $p_{\mathcal{A}}$ be the probability of autoformalizing ψ_0, $\mathcal{A}(\psi_0) = \varphi_1$, s.t. φ_1 is semantically equivalent to

ψ_0, i.e. $\varphi_0 \equiv \varphi_1$. Let p_H be the probability that L hallucinates *FS* φ_1 by autoformalizing ψ_0 s.t. $\varphi_1 \equiv \varphi_0$ even though ψ_0 is not an accurate representation of φ_0.

It can be seen that for a false positive to be output by ∀uto∃∨∧L, the sequence $\varphi_0 \rightarrow \psi_0$ yields an incorrect *NL* description, the sequence $\psi_0 \rightarrow \varphi_1$ autoformalizes incorrectly and hallucinates in the right way to produce $\varphi_1 \equiv \varphi_0$. The probability of such a sequence corresponds to L making two mistakes, with the second mistake being such that it generated an expression equivalent to φ_0. This can be expressed as $(1-p_{\mathcal{I}})(1-p_{\mathcal{A}})p_H$. For $(\mathcal{A}\circ\mathcal{I})^n(\varphi_0)$, this probability is $(1-p_{\mathcal{I}})^n(1-p_{\mathcal{A}})^n p_H^n$ since ∀uto∃∨∧L computes $(\mathcal{A}\circ\mathcal{I})^1(\varphi_i)$ if $\varphi_{i-1} \equiv \varphi_i$ (Sec. 3). As LLM technology improves, we expect $p_{\mathcal{I}}, p_{\mathcal{A}} \rightarrow 1$ and $p_H \rightarrow 0$. As a result, the probability of false positives provided by ∀uto∃∨∧L decreases as n increases. This low likelihood of false positives is further confirmed empirically by our analysis of correlation and predictive power w.r.t. other benchmarks presented above.

§A4: *Can LLMs serve as verifiers?* We use φ_1 generated by GPT-4o (the best performing LLM) and asked an LLM L to evaluate whether $\varphi_0 \equiv \varphi_1$ to check if its output matches that of a formal verifier. It is clear from Fig. 5 that even in this setting (and despite using Chain-of-Thought), LLMs cannot serve as verifiers for anything but toy expressions (low descriptional complexity), after which F_1 scores fall sharply. Our results show that using LLM-as-a-judge is not ideal in applications where truth maintenance is important. We found that LLMs generally struggle to provide correct answers once formulae are larger in general. For smaller expressions, we found that LLMs have difficulties with negations in logic. Due to space limitations, we present an extended analysis of the kinds of syntactic structures that LLMs fail to verify correctly in the Appendix.

5.1 Evaluating Large Reasoning Models (LRMs) using ∀uto∃∨∧L

LRMs are LLMs that also perform some reasoning steps (e.g., search) as a part of their generation process. We evaluated OpenAI's o1, which is the latest LRM available. Due to o1 being 6x more expensive, we regenerated a small dataset with 5 points per category ($|\mathcal{D}| = 200$) and averaged the results across 3 runs with zero-shot prompts. Our results are shown in Fig. 6 and it can be seen that even SOTA LRMs cannot maintain truth effectively in $(\mathcal{A}\circ\mathcal{I})^1(\varphi_0)$.

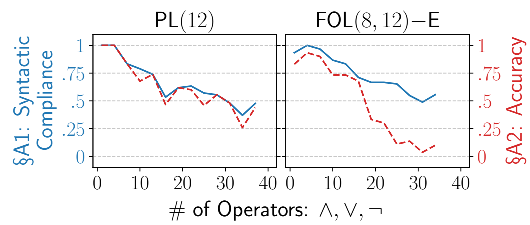


Figure 6: ∀uto∃∨∧L results on OpenAI o1 on a small dataset $|\mathcal{D}| = 200$. Dashed lines indicate accuracy.

6 Related Work

Logical Reasoning RuleTaker (Clark et al., 2020) and ProntoQA (Saparov & He, 2023) generate datasets by using simple "if-then" and syllogisms rules to create reasoning questions. Similar grammars are used by LogicNLI (Tian et al., 2021) and CLUTRR (Sinha et al., 2019). LogiEval (Patel et al., 2024) uses fixed inference rules and LLMs to generate reasoning problems. While these techniques are dynamic, they are limited in their ability to produce interesting reasoning problems across different domains.∀uto∃∨∧L is multi-dimensional providing 5 different datasets, allows multiple customization options, and can generate an infinite number of unique syntax trees.

FOLIO (Han et al., 2022) utilizes human experts to generate a set of reasoning questions based on real-world text sources. They generate questions in both *NL* and *FS* for propositional and first-order logic that require 7 levels of reasoning. A similar approach is employed by ReClor (Yu et al., 2020) and (Srivastava et al., 2023). A key weakness of these approaches is their reliance on human experts..

Autoformalization HumanEval is a popular benchmark for evaluating LLM capabilities of autoformalizing source code. LLM autoformalizations are evaluated via hand-written test cases. It has been shown by Liu et al. (2023a) through the HumanEval+ dataset that the test cases in HumanEval are incomplete and can provide misleading rankings. StructuredRegex (Ye et al., 2020) used crowdsourcing for generating regex datasets. In contrast, ∀uto∃∨∧L requires no human annotations and utilizes formal verifiers for checking the truth maintenance and thus does not share such drawbacks.

FOLIO($\{\mathcal{A}, \mathcal{I}\}$) (Han et al., 2022) tests the (auto/in)formalization abilities of LLMs by using hand-coded annotations of $\langle NL, FS\rangle$ pairs. However, as noted by the authors, they cannot check truth maintenance effectively and rely on an inference engine to compute truth values for each conclusion. ∀uto∃∨∧L uses theorem provers to check equivalence and thus is sound in its accuracy evaluation.

MALLS (Yang et al., 2023) is an autoformalization dataset for first-order logic that was generated using GPT-4. Their use of LLMs for generating the data limits the diversity of the dataset since and the authors suggest to only use this dataset for fine-tuning and not for evaluation. In contrast, ∀uto∃∨∧L generates correct *FS* and has a sound evaluation metric for truth maintenance.

Autoformalization approaches such LeanEuclid (Murphy et al., 2024), DTV (Zhou et al., 2024), LINC (Olausson et al., 2023), SatLM (Ye et al., 2020), Logic-LM (Pan et al., 2023) and others (Wu et al., 2022) utilize formal verifiers to provide sound evaluation metrics but utilize hand-coded datasets that limit their use in evaluating newer LLMs unlike ∀uto∃∨∧L.

Informalization Wu et al. (2022) and ProofNet (Azerbayev et al., 2023) use static datasets to evaluate LLM informalization capabilities. They use metrics such as BLEU scores that are known to not be indicative of accuracy for *FS*-based tasks (Ren et al., 2020). Jiang et al. (2023) develop MMA, a dataset of formal and informal pairs generated using GPT-4. They note that their dataset is an approximate measure due to using LLMs without manual validation. In contrast, ∀uto∃∨∧L is autonomous and provides sound measures of LLM capabilities w.r.t. truth maintenance.

7 Closing Remarks

Conclusions This paper introduced ∀uto∃∨∧L, a new benchmark for autonomous assessment of LLM truth maintenance. Our approach to dataset synthesis allows us to scale without any human labeling. ∀uto∃∨∧L autonomously evaluates $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$ and provides accurate results by using verifiers to guarantee correctness over all inputs. Our framework is easily extensible and provides several prepackaged datasets (and o.o.d. dataset generators) to quickly assess new LLMs. Furthermore, our evaluation indicates that SOTA LLMs and LRMs are not performant in this task. Finally, we show that our metric can be used as an indicator of performance on other *FS*-based tasks and thus can be used as a surrogate benchmark for evaluating new LLMs as and when they are developed.

Broader Impact We introduce a new way to automatically assess whether LLMs can understand their own generations and preserve their truth while automatically scaling datasets. ∀uto∃∨∧L can be used to robustly evaluate the suitability and safety of using LLMs in *FS*-based tasks such as autoformalization, code generation, etc. and can be used as a surrogate to estimate performance when new LLMs are developed. Our work can pave the way for the development of new autonomous techniques for evaluating LLMs in other, more free-structured syntax like conversational AI.

Limitations and Future Work One interesting extension of current work is to utilize the λ-calculus to further expand the datasets that can be generated. Our framework assumes that the generated *NL* uses the English vocabulary. Adding support for other languages is an interesting extension for future work. Another limitation pertains to the use of formal verifiers. It is well-known that first-order logic is undecidable (Huth & Ryan, 2004). We mitigate this by using *FS* verifiers loaded with the appropriate timeout and logging mechanisms (0.66% of our results experienced a timeout). This can be mitigated by using CFGs that generate decidable strings. One interesting application of ∀uto∃∨∧L is to use the generated evaluations as datasets for back-translation to improve the autoformalization capabilities of models (Jiang et al., 2023). Finally, using formal verifiers as tools which the LLM can call is an interesting extension of our benchmark that would further facilitate the assessment of §A4.

Threats to Validity Our reported results for paid APIs are dependent on the model checkpoints used to be available. Similar to existing LLM evaluation methodologies, one must use pass@k and other detailed measures such as std. deviations to increase confidence. We report pass@1 due to the high cost of pass@k for our complete dataset $|\mathcal{D}^*| \sim 85k \times 2$ prompts but do report std. deviations across 10 runs (on a single batch of $2k$ samples) in App. H. As is the case with all approaches, our approach assumes the soundness of the verifier programs and parsing libraries used. Software bugs in the verifier program or parsing libraries could cause false signals to be output by ∀uto∃∨∧L. Our use of open-source popular libraries such as Prover9, NLTK reduces our exposure to such risk.

Ethical Considerations Our work involves using LLMs for generating text. Naturally, it is imperative to ensure that appropriate guardrails are in place to prevent offensive content from being generated and/or displayed. We do not use any personally identifiable information in ∀uto∃∨∧L.

References

Reg2dfa. https://github.com/Jack-Q/reg2dfa, 2017. Accessed: 2024-06-01.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *CoRR*, abs/2302.12433, 2023.

Satanjeev Banerjee and Alon Lavie. 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*, pp. 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W05-0909.

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly, 2009.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the role of Bleu in machine translation research. In *Proc. EACL*, 2006.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

Erzsébet Csuhaj-Varjú and Alica Kelemenová. Descriptional complexity of context-free grammar forms. *Theoretical Computer Science*, 112(2):277–289, 1993.

Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: An efficient SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2008.

TJ Dunham and Henry Syahputra. Reactor mk. 1 performances: Mmlu, humaneval and bbh test results. *arXiv preprint arXiv:2406.10515*, 2024.

Daniele Faraglia. Faker. https://arhttps://github.com/joke2k/faker, 2024. Accessed: 2023-06-01.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard, 2024.

IBM Granite Team. Granite 3.0 language models, 2024.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094, 2023.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. FOLIO: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XPZIaotutsD.

J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley series in computer science. Addison-Wesley, 2001. ISBN 9780201441246.

Michael Huth and Mark Dermot Ryan. *Logic in Computer Science - Modelling and Reasoning about Systems*. Cambridge University Press, 2nd edition, 2004.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Albert Q. Jiang, Wenda Li, and Mateja Jamnik. Multilingual mathematical autoformalization. *CoRR*, abs/2311.03755, 2023.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W04-1013`.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *Conf. NeurIPS*, 2023a.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

William McCune. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2010.

Microsoft. Phi-3 technical report: A highly capable language model locally on your phone. `https://arxiv.org/pdf/2404.14219`, 2024. Accessed: 2023-06-01.

MistralAI. Introducing the world's best edge models. `https://mistral.ai/news/ministraux/`, 2024. [Accessed 22-11-2024].

Logan Murphy, Kaiyu Yang, Jialiang Sun, Zhaoyu Li, Anima Anandkumar, and Xujie Si. Autoformalizing euclidean geometry. In *Proc. ICML*, 2024.

Theo Olausson, Alex Gu, Benjamin Lipkin, Cedegao E. Zhang, Armando Solar-Lezama, Joshua B. Tenenbaum, and Roger Levy. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proc. EMNLP*, 2023.

OpenAI. Gpt-4o. `https://arxiv.org/pdf/2303.08774.pdf`, 2023. Accessed: 2023-06-01.

OpenAI. Gpt-4o-mini. `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/`, 2024. Accessed: 2024-09-29.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *EMNLP Findings*, 2023.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. Multi-logieval: Towards evaluating multi-step logical reasoning ability of large language models. *arXiv preprint arXiv:2406.17169*, 2024.

Qwen2. Qwen 2.5. `https://qwen2.org/qwen2-5/`, 2024. [Accessed 22-11-2024].

Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. Codebleu: a method for automatic evaluation of code synthesis. *CoRR*, 2020.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2020.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations (ICLR)*, 2023.

Karin Kipper Schuler. *VerbNet: A Broad-coverage, Comprehensive Verb Lexicon*. PhD thesis, University of Pennsylvania, 2005. AAI3179808.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. Diagnosing the first-order logical reasoning ability through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Cheng Xu, Shuhao Guan, Derek Greene, and M-Tahar Kechadi. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*, 2024.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. Harnessing the power of large language models for natural language to first-order logic translation. *CoRR*, abs/2305.15541, 2023. doi: 10.48550/ARXIV.2305.15541.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Benchmarking multimodal regex synthesis with complex structures. In *Proc. ACL*, 2020.

Yi. 01-ai. `https://huggingface.co/01-ai/Yi-1.5-34B-Chat`, 2024. Accessed: 2024-11-22.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. ReClor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations (ICLR)*, 2020.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SkeHuCVFDr`.

Jin Peng Zhou, Charles Staats, Wenda Li, Christian Szegedy, Kilian Q. Weinberger, and Yuhuai Wu. Don't trust: Verify - grounding LLM quantitative reasoning with autoformalization. In *Proc. ICLR 2024*, 2024.

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 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 ∀uto∃∨∧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, ∀uto∃∨∧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 $\overline{\varphi}$
3: $\mathcal{N} \leftarrow \{0 : [\textit{None}]\}, \mathcal{N}_t \leftarrow \langle\rangle$
4: **for** $d = 1, 2, \ldots, \textit{depth}$ **do**
5:     $\mathcal{N}' \leftarrow \textit{sampleN}(\mathcal{N}[d-1], n)$
6:     **for** $\nu \in \mathcal{N}'$ **do**
7:         $\mathcal{N}_\nu, \mathcal{T}_\nu \leftarrow \textit{generateNChildren}(\nu, \mathcal{G}, n)$
8:         $\mathcal{N}[d] \mathrel{+}= \mathcal{N}_\nu$
9:         $\mathcal{N}_t \leftarrow \mathcal{N}_t \cup \mathcal{T}_\nu$
10:     **end for**
11: **end for**
12: $M \leftarrow \textit{categorizeExpressionsIntoDict}(\mathcal{N}_t, m)$
13: $\overline{\varphi} \leftarrow \langle\rangle$
14: **for** $k \in keys(M)$ **do**
15:     $M_k \leftarrow \textit{sampleCFGExpressions}(M[k], \textit{sample\_count})$
16:     $\overline{\varphi}_k \leftarrow \textit{buildFSExpressions}(M_k, \mathcal{V})$
17:     $\overline{\varphi} \leftarrow \overline{\varphi} \cup \overline{\varphi}_k$
18: **end for**
19: **Return:** $\overline{\varphi}$

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 φ be an FS expression that can be constructed using the rules from CFG $\mathcal{G}$ and the vocabulary $\mathcal{V}$. Note that φ corresponds to a CFG expression φ_{CFG}, derived by substituting the vocabulary with the CFG symbols. Due to uniform selection, the probability of φ being generated from φ_{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, φ 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.

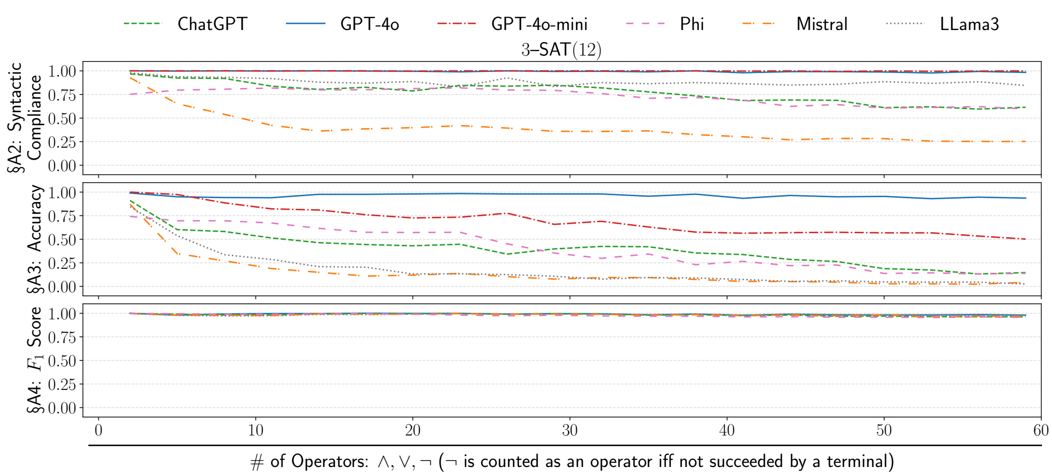


Figure 7: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) for 3-SAT from using ∀uto∃∨∧L to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the # of operators.

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
General	`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.
First-Order Logic	`free_variable_prob`	0.25	Probability of a constant being replaced by a variable.
	`max_free_variables`	∞	Maximum number of unique variables.
	`max_predicate_arity`	2	Maximum predicate arity.
	`min_predicate_arit`	1	Minimum predicate arity.
	`num_objects`	12	Number of unique constants.
	`num_predicates`	8	Number of unique predicates.
Propositional Logic	`num_propositions`	12	Number of unique propositions.
Regular Expression	`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 ∀uto∃∨∧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.

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

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.

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
$\forall < x1 >< x2 >$ ... $< xn >$. represents universal quantification with x1... representing free variables
$\exists < x1 >< x2 >$ ... $< xn >$. represents existential quantification with x1... 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 $\forall$uto$\exists\vee\wedge$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 were 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 formluae in $\forall$uto$\exists\vee\wedge$L. 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 ∀uto∃∨∧L for the propositional logic dataset.

φ	$\mathcal{I}(\varphi)$	$(\mathcal{A} \circ \mathcal{I})(\varphi)$
$(\neg p11 \wedge \neg p8)$	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 ∀uto∃∨∧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 ∀uto∃∨∧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 ∀uto∃∨∧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 ∀uto∃∨∧L for the first-order logic dataset.

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

Table 5: Examples of errors the evaluated LLMs made while evaluating with ∀uto∃∨∧L for the regular expression dataset.

φ	$\mathcal{I}(\varphi)$	$(\mathcal{A} \circ \mathcal{I})(\varphi)$
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*1*1*
(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 ∀uto∃∨∧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 ($\land, \lor, \neg$). 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.

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

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

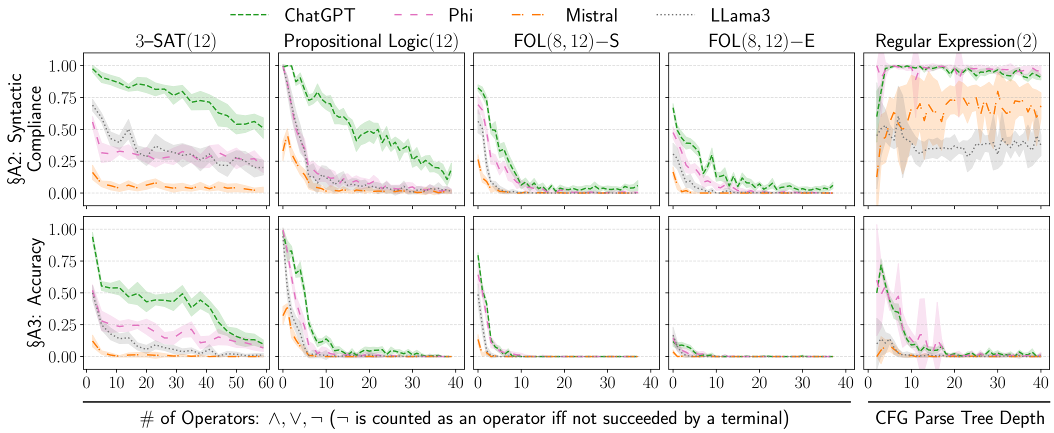


Figure 8: Average and standard deviation error of Zero-shot Pass@1 results from using ∀uto∃∨∧L to assess LLMs w.r.t. §A2 and §A3 (Sec. 4) on the first batch of the packaged datasets. The x-axis represents an increasing order of descriptional complexity.

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.

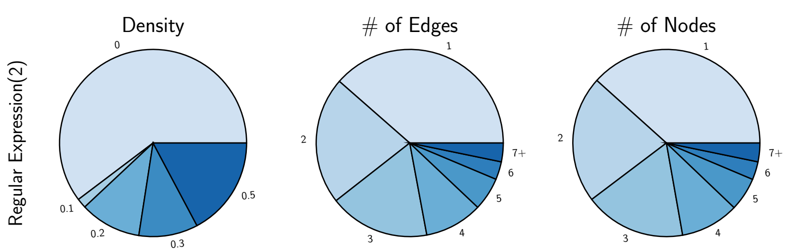


Figure 9: Count of the number of examples for each metric value for the regular expression datasets. The pie charts increase in values counter-clockwise while going from lighter to darker.

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.

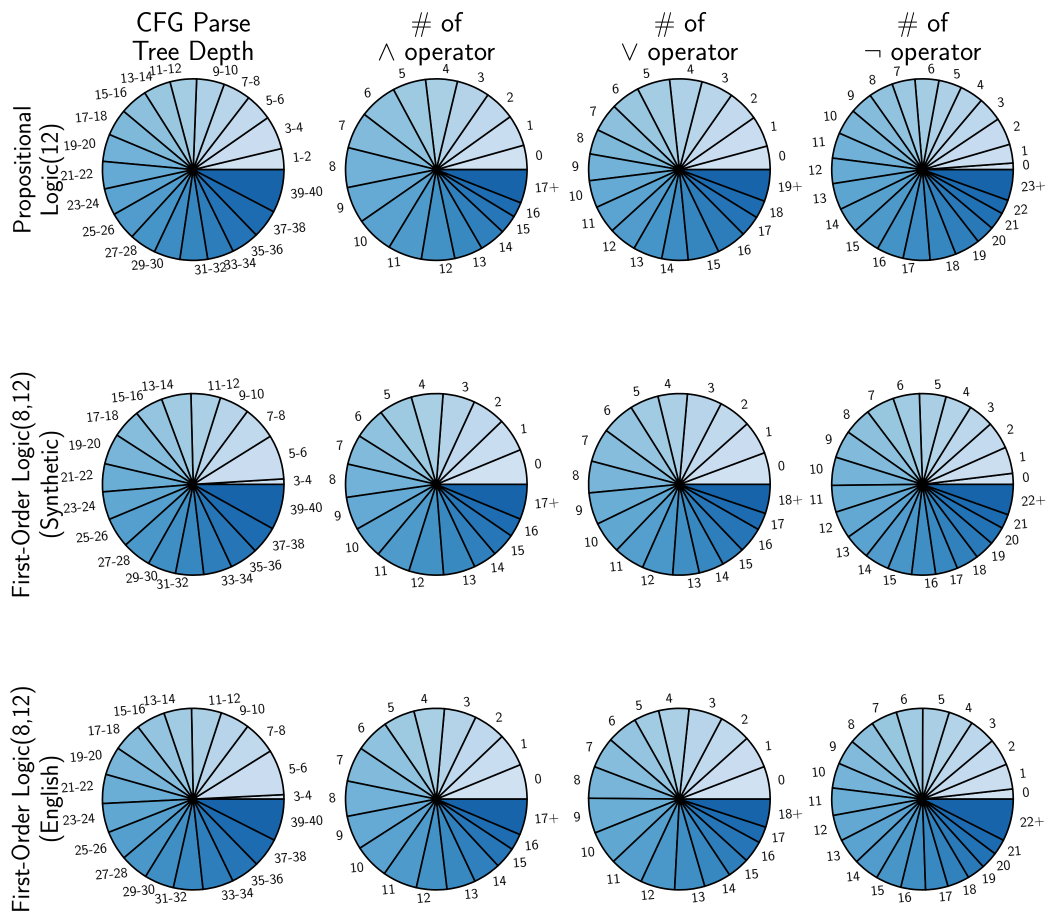


Figure 10: Count of the number of examples for each metric value for each of the datasets. Each row is a dataset and each column is a different metric that can be used to categorize the dataset. The pie charts increase in value counter-clockwise while going from lighter to darker.

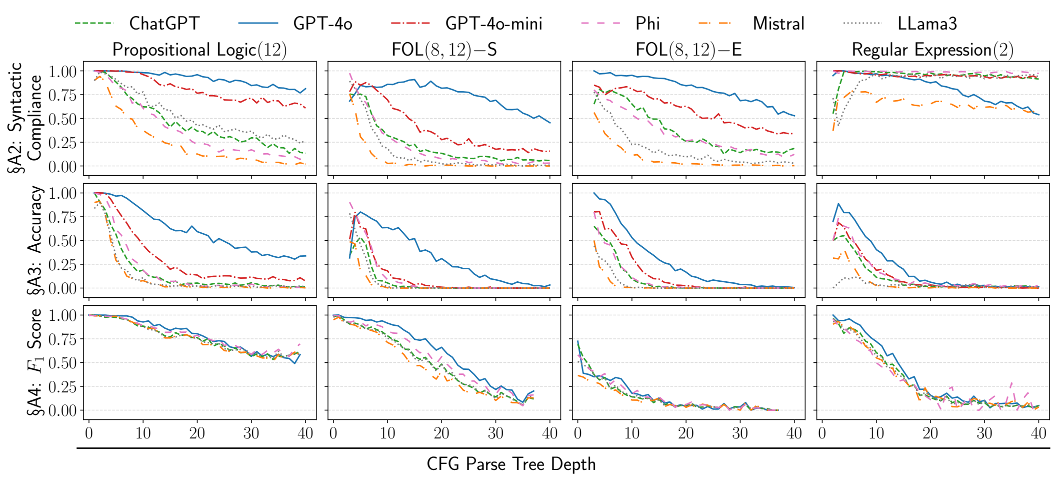


Figure 11: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using ∀uto∃∨∧L to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the depth of the CFG tree to produce the formula.

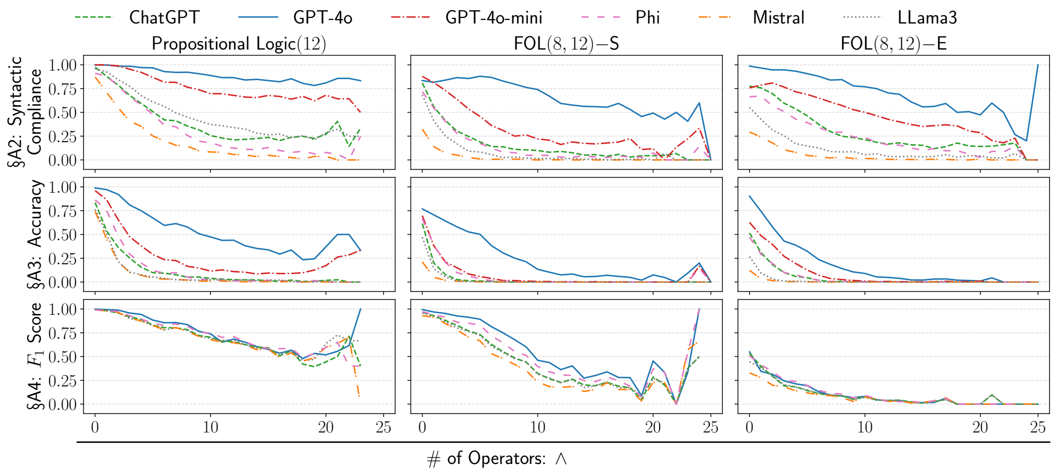


Figure 12: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using ∀uto∃∨∧L to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the number of and operators (∧) in the expression.

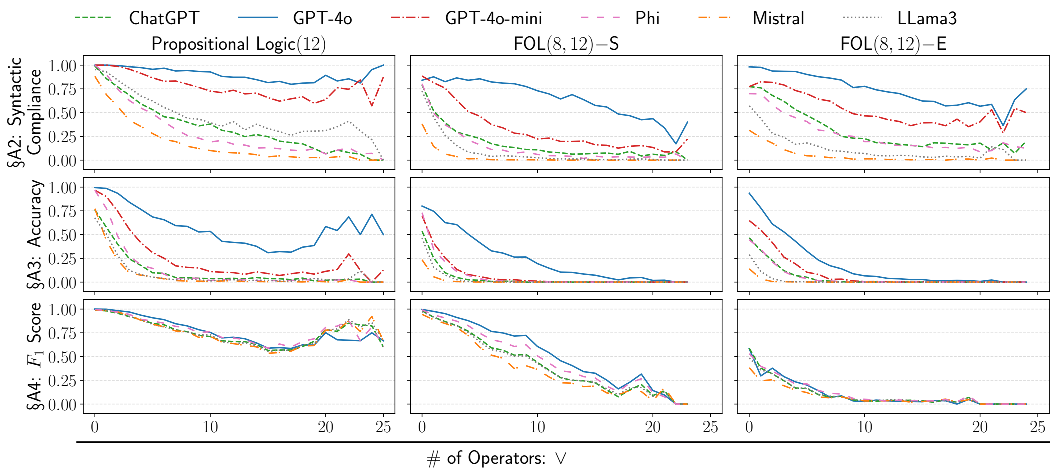


Figure 13: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using ∀uto∃∨∧L to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the number of or operators (∨) in the expression.

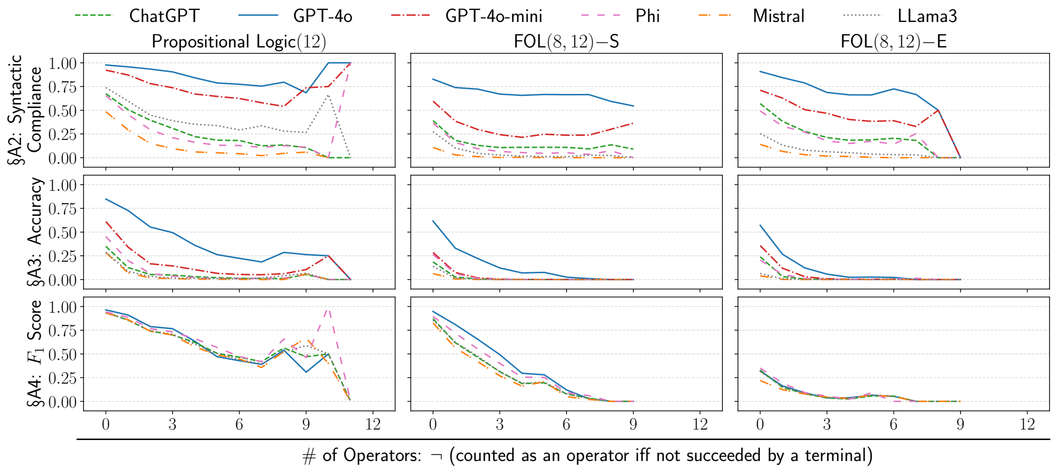


Figure 14: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using $\forall$uto$\exists\vee\wedge$L to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the number of negation operators ($\neg$) in the expression.

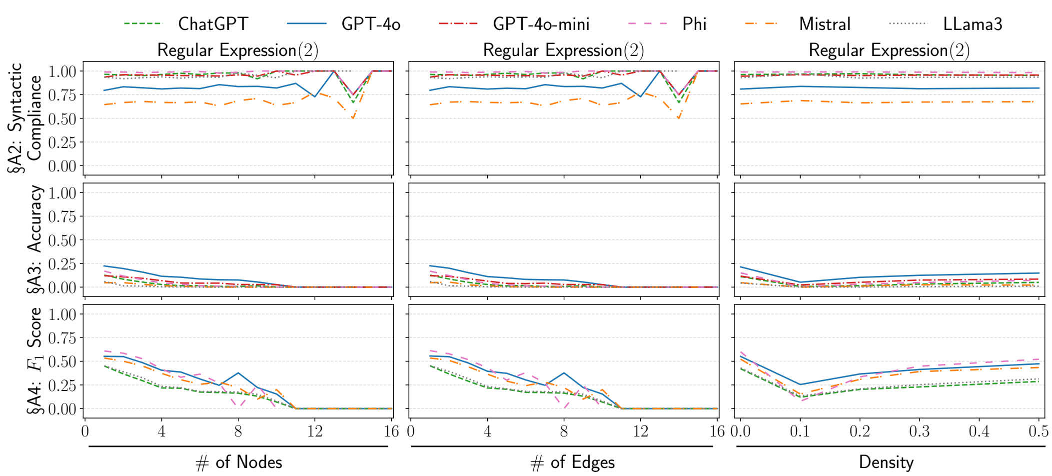


Figure 15: Zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using $\forall$uto$\exists\vee\wedge$L to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis is the metric on the CFG tree to produce the regular expression formula.

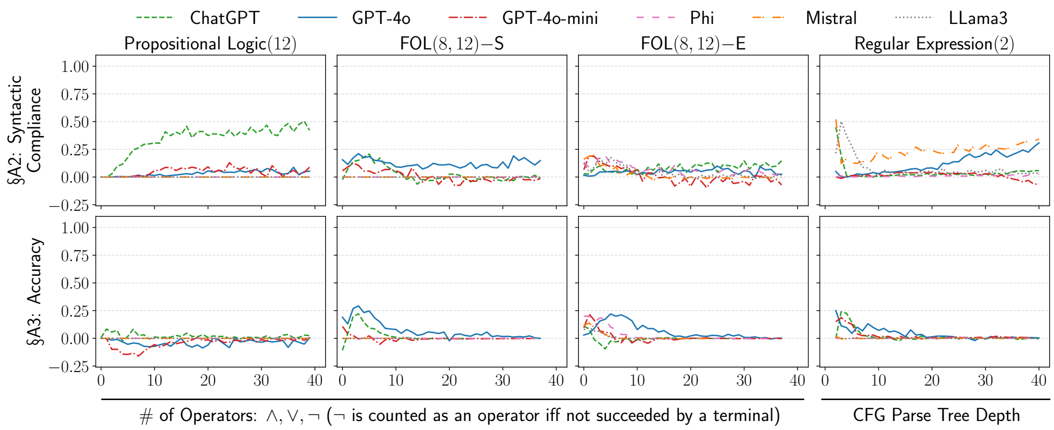


Figure 16: Syntactic compliance and accuracy difference of few-shot Pass@1 compared to zero-shot Pass@1 results (avg. over 10 batches, higher values better) from using ∀uto∃∨∧L to assess LLMs w.r.t. §A2, §A3, §A4 (Sec. 4) on the packaged datasets. The x-axis represents the increasing order of descriptional 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 ∀UTO∃∨∧L PREDICTIVE POWER EVALUATION

For evaluating the correlation between a LLM's performance on ∀uto∃∨∧L and existing benchmarks and measuring the predictive power of ∀uto∃∨∧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 ∀uto∃∨∧L.

In this section, we discuss the experimental setup for the benchmark, the sources used for LLM performance on other benchmarks, and the ∀uto∃∨∧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]
$\forall x(DrinkRegularly(x, coffee) \lor (\neg WantToBeAddictedTo(x, caffeine)))$
People regularly drink coffee, or they don't want to be addicted to caffeine, or both.
[VOCABULARY]
$\lor$ represents disjunction
$\land$ represents conjunction
$\neg$ represents negation
$\rightarrow$ represents implication
(and) represent parentheses
propositions can be used verbatim
predicates can be used verbatim
$\forall < x1 >< x2 > ... < xn >$. represents universal quantification with x1... representing free variables
$\exists < x1 >< x2 > ... < xn >$. represents existential quantification with x1... representing free variables
The objects are: caffeine
The parameterized predicates are: $awarethatdrug(?p0, ?p1)$, $wanttobeaddictedto(?p0, ?p1)$
The free variables are: x

[FORMULA]
$\forall x.\ (\neg wanttobeaddictedto(x, caffeine) \rightarrow \neg awarethatdrug(x, caffeine))$

For the FOLIO premise informalization and autoformalization experiments, the LLM was prompted using the same few-shot first-order logic prompt used by ∀uto∃∨∧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 answered 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-LogicEval (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 disjunction
Use $\wedge$ to represent conjunction
Use $\neg$ 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 HumanEval and Big Bench Hard Score Sources

To evaluate the correlation and predictive power of ∀uto∃∨∧L 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"

Table 6: Reported performance of SOTA LLMs on HumanEval and Big Bench Hard (BBH) benchmarks. The values under the Computed column are averaged over 5 runs from our experiments. Other results are reported from online sources. A – indicates that we were not able to find any online source. We used our local computed results when they were available.

Model	HumanEval Score		BBH Score
	Computed	(Online)	(Online)
ChatGPT	74.3	68 (OpenAI, 2024)	48.1 (OpenAI, 2023)
GPT-4o	91.8	90.2 (OpenAI, 2024)	48.1 (Dunham & Syahputra, 2024)
GPT-4o-mini	88.3	87.2 (OpenAI, 2024)	–
Llama-3.2-1B-Instruct	34.6	–	8.7 (Fourrier et al., 2024)
Qwen-2.5-1.5B-Instruct	56.7	61.6 (Qwen2, 2024)	19.8 (Fourrier et al., 2024)
Phi-3.5-Mini-Instruct	71.3	64.6 (Liu et al., 2023b)	36.7 (Fourrier et al., 2024)
Mistral-7B-Instruct-v0.2	44.5	42.1 (Liu et al., 2023b)	24.0 (Fourrier et al., 2024)
Llama-3-8B-Instruct	62.8	61.6 (Liu et al., 2023b)	24.5 (Fourrier et al., 2024)
Granite-3.0-8B-Instruct	62.2	64.6 (Granite Team, 2024)	51.6 (Fourrier et al., 2024)
Llama-3.1-8B-Instruct	63.4	66.5 (Microsoft, 2024)	63.4 (Microsoft, 2024)
Ministral-8B-Instruct-2410	76.8	76.8 (MistralAI, 2024)	8.7 (Fourrier et al., 2024)
Gemma-2-9B-IT	68.3	68.9 (Qwen2, 2024)	42.1 (Fourrier et al., 2024)
Phi-3-Medium-4k-Instruct	75.0	62.2 (Microsoft, 2024)	49.4 (Fourrier et al., 2024)
Qwen-2.5-14B-Instruct	80.5	83.5 (Qwen2, 2024)	48.4 (Fourrier et al., 2024)
Yi-1.5-34B-Instruct	72.6	75.2 (Yi, 2024)	44.3 (Fourrier et al., 2024)
Llama-3-70B-Instruct	79.9	77.4 (Liu et al., 2023b)	50.2 (Fourrier et al., 2024)

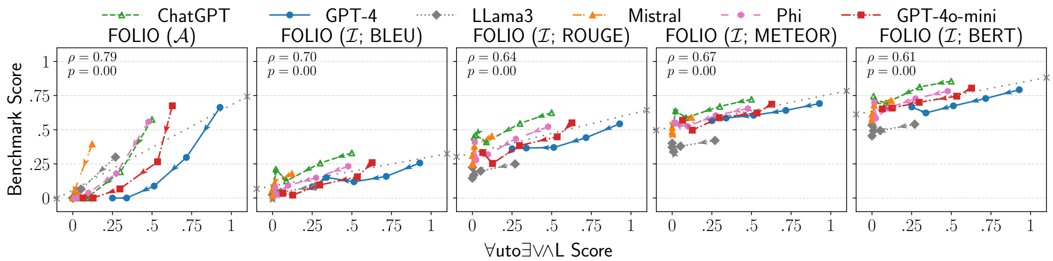


Figure 17: Correlation between scores on ∀uto∃∨∧L and both autoformalization $\mathcal{A}$ and informalization $\mathcal{I}$ for FOLIO premises. Each point represents a specific number of operators with arrows showing increasing complexity (number of operators). The trendline across all the points is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

K.4 Computed ∀uto∃∨∧L Conditional Performance

To compare against the performance on different benchmarks in Section 5, we needed to calculate the conditional performance of each LLM on ∀uto∃∨∧L 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 of the first-order logic formulae with less than seven operators when calculating the correlation and predictive power. On MultiLogiEval, the number of operators is 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 ∀uto∃∨∧L. 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 compare to the accuracy of the same number of operators in the first-order logic dataset we generated.

We plot these results in Figure 17 with the Pearson correlation 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 ∀uto∃∨∧L, autoformalization, and informalization decreases. The autoformalization correlation is significantly stronger due to the informalization evaluation metrics being much weaker at evaluating truth maintenance.

L LLM as Verifiers Evaluation

In this section, we analyze the performance of LLMs on §A4, where we evaluate the performance of using a LLM to verify whether the formal syntax expression φ is equivalent to the one produced by GPT-4o after doing $\mathcal{A} \circ \mathcal{I}(\varphi)$. Figure 20 shows the number of positive and negative examples. Additionally, it breaks down the performance of each LLM on each dataset.

The LLMs are clearly biased towards giving positive answers with a high ratio of false positives to negative labels compared to false negatives to positive labels. A common case was the LLM not recognizing that GPT-4o renamed predicates or constants producing a different formal syntax expression. A structure that ChatGPT tends to struggle with is double negations. Below we provide two examples where, even at the time of writing this rebuttal, ChatGPT failed to correctly verify whether the two formulae are equivalent. Note that [FORMULA 2] is ChatGPT's own response after conducting $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$ where $\varphi_0 =$ [FORMULA 1].

Output Example 1: Regular Expression Verification Example

Prompt:
Your task is to say whether two Regular Expression 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]: 100*
[FORMULA 2]: 1(0*)

Response:
To determine whether the two regular expressions $100*$ and $1(0*)$ are equivalent, let's analyze them:
Explanation:
1. **Formula 1: 100***
- This matches the character "1" followed by zero or more "0"s.
- It will match: "1", "10", "100", "1000", etc.
2.**Formula 2: 1(0*)**
- This matches the character "1" followed by zero or more "0"s (the same as Formula 1).
- It will also match: "1", "10", "100", "1000", etc.
Key Observation:
- Both formulas describe the same set of strings: a "1" followed by zero or more "0"s.
- The use of parentheses in Formula 2 is purely for grouping and does not alter the matching behavior.
Conclusion:
The two regular expressions are equivalent.
[Answer]: yes

Output Example 2: First-Order Logic Verification Example

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]: $\forall x1.(\neg\neg pred8(p8, p7) \vee \neg pred4(x1))$
[FORMULA 2]: $\forall x1.(\neg pred8(p8, p7) \vee \neg pred4(x1))$

Response:
The two formulas are equivalent.
Explanation: In both Formula 1 and Formula 2, the part ¬¬pred8(p8, p7) is equivalent to ¬pred8(p8, p7) due to the double negation elimination rule in logic. Therefore, both formulas convey the same meaning, as the presence or absence of a double negation does not affect the truth value in this context.
[Answer]: Yes

M Dataset Diversity

Fig. 9 and Fig. 9 provide additional details on the types of data present in the datasets packaged with ∀uto∃∨∧L. Users can generate dynamic datasets along these dimensions using the hyperparameters mentioned in Table 1.

To further provide additional statistics pertaining to the similarity of formulae in our dataset, especially those where the formulae are otherwise equivalent but just use different vocabularies. For example, the formula $f = p_1$ can be represented via different propositions where $p_1 =$ *It is raining* in f_1 and something different in another formula f_2 even though they canonically represent the same formula f.

This allows to test robustness in LLM outputs. Nevertheless, the probability of such instances decreases as the formula size increases. We have counted the total proporition of the dataset where this occurs by replacing any variable from the vocabulary with an element of a vocabulary of size 1. For example, all variables used in PL(12) dataset of our results are replaced by substituting those variables with a vocabulary of only 1 proposition. Excess parentheses etc are preprocessed using NLTK and removed before the substitution (e.g. $((p_1) \wedge p_2)$ is simplified to $p_1 \wedge p_2$.

The k-SAT dataset contains 8550 unique samples and the propositional logic dataset contains 17.7k samples constituting 85% and 90% of these datasets respectively.

N EVALUATION OF LLMS

Table 7 lists the models, their parameters (– if closed-source), and the exact model version used for our experiments. The open-source models were loaded using NVIDIA A100 80GB GPUs whereas we used the OpenAI API for the GPT family of models. We cover a diverse range of models in our evaluation ranging from extremely small LMs with a few billion parameters ($\sim$1B) to LLMs with several billions of parameters. This allows the analysis of ∀uto∃∨∧L from the lens of generalization.

Fig. 18 represents the syntactic compliance (§A2) data from Fig. 5 for all the models with a separate axis for each LLM. Similarly, Fig. 19 plots the Accuracy (§A3). Tables 9 – 14 provide the data that was used to plot the results in Fig. 3 and to compute the predictive power in Fig. 4.

N.1 CLAUDE EVALUATION

We evaluated Claude 3.0 Sonnet on just the 3-SAT, propositional logic, and regular expression datasets due to the cost. Our results are shown in Figure 21 and show that Claude 3.0 Sonnet performs similarly to GPT-4o with both having nearly perfect syntactic compliance and accuracy on 3-SAT. Sonnet achieved the highest syntactic compliance and accuracy on propositional logic compared to the other models. However the accuracy was only around 50% for expressions with more than 20 operators. Additionally, while being often syntactic compliant, Sonnet performed with low accuracy on the regular expression dataset.

Table 7: The LLMs used in our evaluation. The label names represent the labels used in Fig. 18 and Fig. 19, $|\theta|$ represents the total number of parameters, and the last column lists the exact version used (for reproducibility).

Label	$\|\theta\|$	**Version**
ChatGPT	–	`GPT-3.5-turbo-0125`
GPT-4o	–	`gpt-4o-2024-08-06`
GPT-4o-mini	–	`gpt-4o-mini-2024-07-18`
GPT-4o1	–	`o1-preview-2024-09-12`
Llama-3.2-1B-Instruct	1B	`meta-llama/Llama-3.2-1B-Instruct`
Qwen-2.5-1.5B-Instruct	1.5B	`Qwen/Qwen2.5-1.5B-Instruct`
Phi-3.5-Mini-Instruct	4B	`microsoft/Phi-3.5-mini-instruct`
Mistral-7B-Instruct-v0.2	7B	`mistralai/Mistral-7B-Instruct-v0.2`
Llama-3-8B-Instruct	8B	`meta-llama/Llama-3-8B-Instruct`
Granite-3.0-8B-Instruct	8B	`ibm-granite/granite-3.0-8b-instruct`
LLama-3.1-8B-Instruct	8B	`meta-llama/Llama-3.1-8B-Instruct`
Ministral-8B-Instruct-2410	8B	`mistralai/Ministral-8B-Instruct-2410`
Gemma-2-9B-IT	9B	`google/gemma-2-9b-it`
Phi-3-Medium-4k-Instruct	14B	`microsoft/Phi-3-medium-4k-instruct`
Qwen-2.5-14B-Instruct	14B	`Qwen/Qwen2.5-14B-Instruct`
Yi-1.5-34B-Instruct	34B	`01-ai/Yi-34B-Instruct`
Llama-3-70B-Instruct	70B	`meta-llama/Llama-3-70B-Instruct`

Table 8: Correlation data for FOLIO(R; NL). The ∀uto∃∨∧L data was averaged from the PL dataset with data points with description complexity $d \leq 6$. These values were used to compute the predictive power of ∀uto∃∨∧L reported in Fig. 4.

Model	∀uto∃∨∧L Score	FOLIO(R; NL) Score
GPT-4o	0.79	0.75
GPT-4o-mini	0.56	0.69
ChatGPT	0.36	0.56
Mistral-7B-Instruct-v0.2	0.06	0.54
Phi-3-medium-4k-insruct	0.35	0.67
LLama-3-8B-Instruct	0.13	0.58
Gemma-2-9B-IT	0.28	0.64
Granite-3.0-8B-Instruct	0.18	0.60
Llama-3.1-8B-Instruct	0.09	0.59
LLama-3.2-1B-Instruct	0.03	0.36
LLama-3-70B-Instruct	0.49	0.70
Ministral-8B-Instruct-2410	0.10	0.61
Phi-3.5-Mini-Instruct	0.19	0.61
Qwen-2.5-1.5B-Instruct	0.07	0.49
Qwen-2.5-14B-Instruct	0.67	0.73
Yi-1.5-34B	0.21	0.63

Table 9: Correlation data for FOLIO(R; FOL). The ∀uto∃∨∧L data was averaged from the FOL dataset with data points with description complexity $d \leq 6$. These values were used to compute the predictive power of ∀uto∃∨∧L reported in Fig. 4.

Model	∀uto∃∨∧L Score	FOLIO(R; FOL) Score
GPT-4o	0.79	0.71
GPT-4o-mini	0.56	0.67
ChatGPT	0.36	0.51
Mistral-7B-Instruct-v0.2	0.06	0.51
Phi-3-medium-4k-insruct	0.35	0.62
LLama-3-8B-Instruct	0.13	0.52
Gemma-2-9B-IT	0.28	0.59
Granite-3.0-8B-Instruct	0.18	0.56
Llama-3.1-8B-Instruct	0.09	0.56
LLama-3.2-1B-Instruct	0.03	0.36
LLama-3-70B-Instruct	0.49	0.66
Ministral-8B-Instruct-2410	0.10	0.56
Phi-3.5-Mini-Instruct	0.19	0.53
Qwen-2.5-1.5B-Instruct	0.07	0.45
Qwen-2.5-14B-Instruct	0.67	0.71
Yi-1.5-34B	0.21	0.61

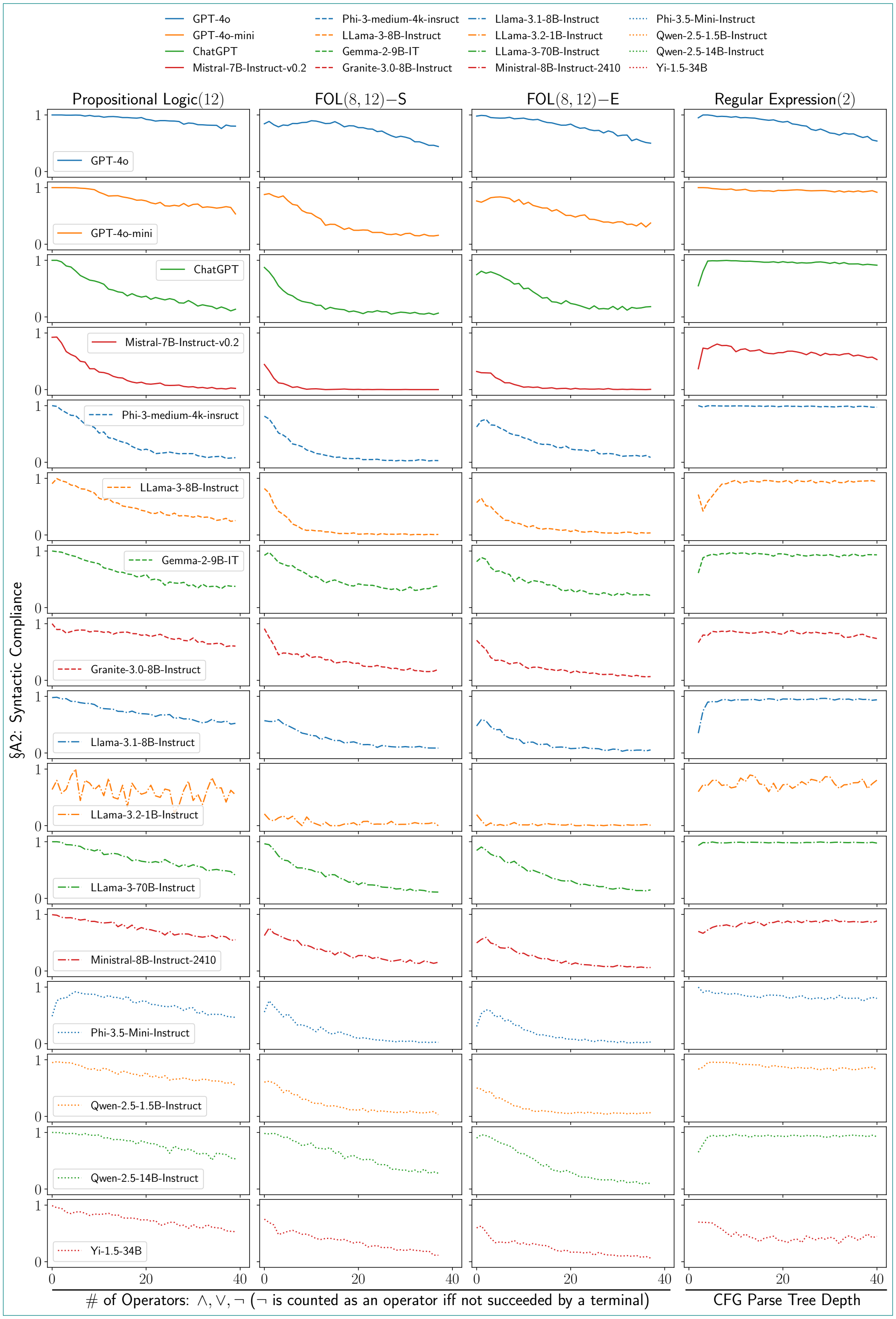


Figure 18: Syntactic compliance (§A2) of all models on the ∀uto∃∨∧L datasets.

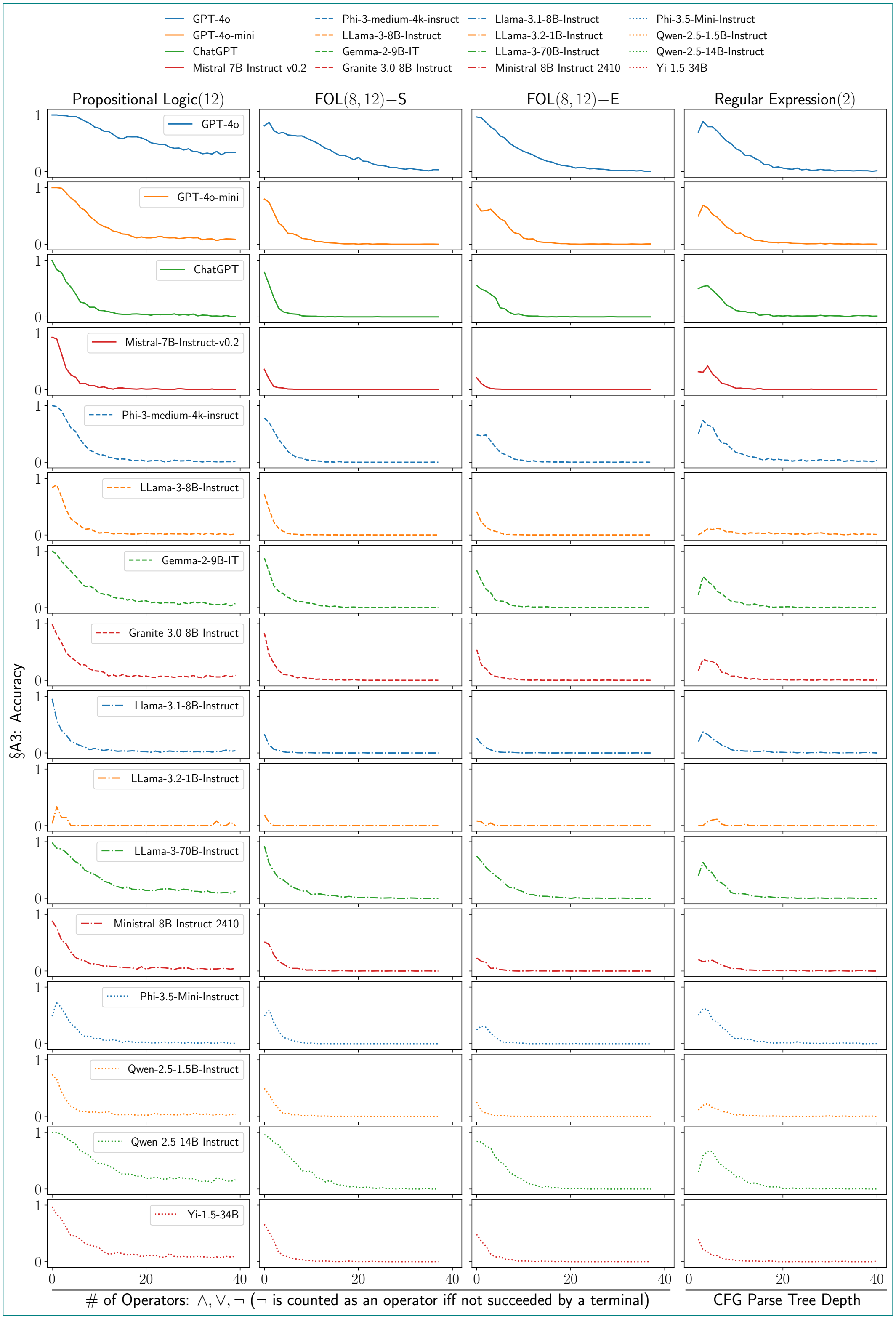


Figure 19: Accuracy (§A3) of all models on the ∀uto∃∨∧L datasets.

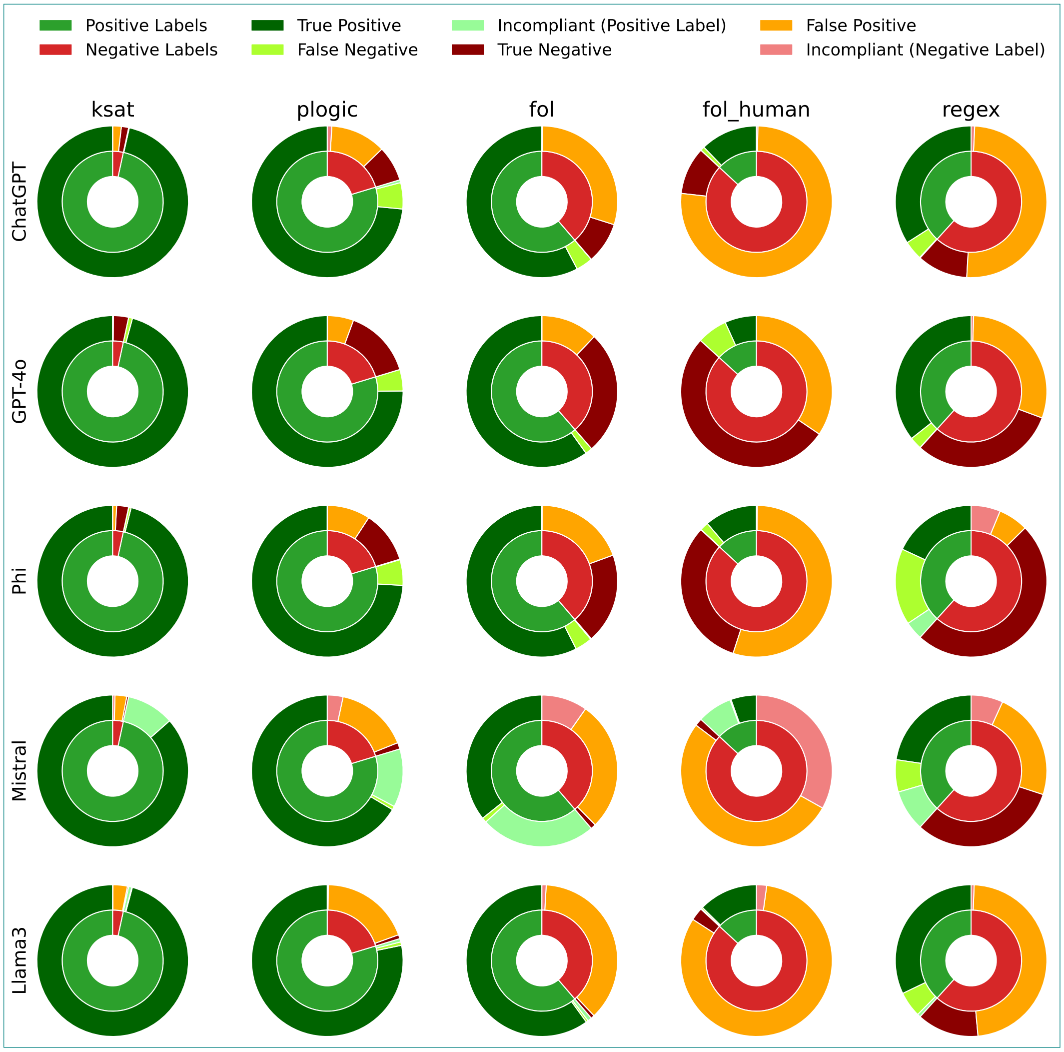


Figure 20: The number of positive and negative examples of $\varphi_1 \equiv \mathcal{A} \circ \mathcal{I}(\varphi_0)$ when evaluating GPT-4o on ∀uto∃∨∧L for each dataset (inner donuts). Additionally included is a breakdown of the performance of each LLM when acting as the verifier (outer donuts). Included are all examples containing 20 or fewer operators or, in the case of the regular expression dataset, CFG tree depth of 20 or fewer. Incompliant represents syntactically incompliant φ_1 generated by GPT-4o for ground-truth φ_0 via $(\mathcal{A} \circ \mathcal{I})^n(\varphi_0)$

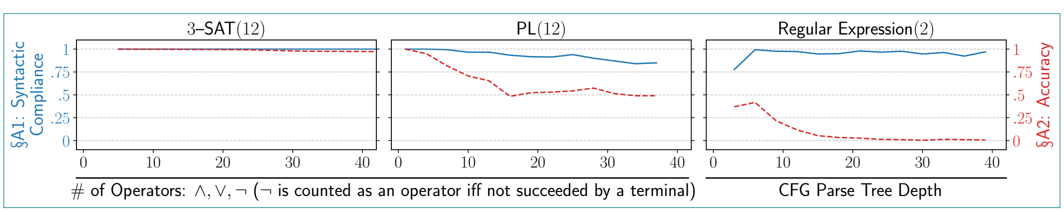


Figure 21: ∀uto∃∨∧L results on Claude 3.0 Sonnet on the 3-SAT, propositional logic, and regular expression datasets. Dashed line is the accuracy.

Table 10: Correlation data for LogiEval(R; PL). The ∀uto∃∨∧L data was averaged from the PL dataset with data points with description complexity $d \leq 30$. These values were used to compute the predictive power of ∀uto∃∨∧L reported in Fig. 4.

Model	∀uto∃∨∧L Score	LogiEval(R; PL) Score
GPT-4o	0.67	0.87
GPT-4o-mini	0.35	0.67
ChatGPT	0.17	0.64
Mistral-7B-Instruct-v0.2	0.12	0.60
Phi-3-medium-4k-insruct	0.23	0.75
LLama-3-8B-Instruct	0.12	0.61
Gemma-2-9B-IT	0.28	0.71
Granite-3.0-8B-Instruct	0.21	0.58
Llama-3.1-8B-Instruct	0.11	0.71
LLama-3.2-1B-Instruct	0.04	0.50
LLama-3-70B-Instruct	0.34	0.85
Ministral-8B-Instruct-2410	0.17	0.68
Phi-3.5-Mini-Instruct	0.10	0.62
Qwen-2.5-1.5B-Instruct	0.11	0.52
Qwen-2.5-14B-Instruct	0.46	0.76
Yi-1.5-34B	0.26	0.78

Table 11: Correlation data for LogiEval(R; FOL). The ∀uto∃∨∧L data was averaged from the FOL dataset with data points with description complexity $d \leq 30$. These values were used to compute the predictive power of ∀uto∃∨∧L reported in Fig. 4.

Model	∀uto∃∨∧L Score	LogiEval(R; FOL) Score
GPT-4o	0.32	0.82
GPT-4o-mini	0.17	0.56
ChatGPT	0.09	0.63
Mistral-7B-Instruct-v0.2	0.01	0.56
Phi-3-medium-4k-insruct	0.09	0.70
LLama-3-8B-Instruct	0.02	0.62
Gemma-2-9B-IT	0.07	0.69
Granite-3.0-8B-Instruct	0.05	0.55
Llama-3.1-8B-Instruct	0.02	0.68
LLama-3.2-1B-Instruct	0.00	0.47
LLama-3-70B-Instruct	0.15	0.78
Ministral-8B-Instruct-2410	0.02	0.64
Phi-3.5-Mini-Instruct	0.04	0.54
Qwen-2.5-1.5B-Instruct	0.02	0.50
Qwen-2.5-14B-Instruct	0.19	0.66
Yi-1.5-34B	0.05	0.71

Table 12: Correlation data for FOLIO($\mathcal{A}$). The ∀uto∃∨∧L data was averaged from the FOL dataset with data points with description complexity $d \leq 6$. These values were used to compute the predictive power of ∀uto∃∨∧L reported in Fig. 4.

Model	∀uto∃∨∧L Score	FOLIO($\mathcal{A}$) Score
GPT-4o	0.82	0.48
GPT-4o-mini	0.58	0.47
ChatGPT	0.40	0.38
Mistral-7B-Instruct-v0.2	0.07	0.23
Phi-3-medium-4k-insruct	0.38	0.37
LLama-3-8B-Instruct	0.15	0.18
Gemma-2-9B-IT	0.32	0.35
Granite-3.0-8B-Instruct	0.21	0.16
Llama-3.1-8B-Instruct	0.10	0.26
LLama-3.2-1B-Instruct	0.03	0.00
LLama-3-70B-Instruct	0.53	0.47
Ministral-8B-Instruct-2410	0.11	0.26
Phi-3.5-Mini-Instruct	0.22	0.21
Qwen-2.5-1.5B-Instruct	0.08	0.12
Qwen-2.5-14B-Instruct	0.71	0.42
Yi-1.5-34B	0.24	0.37

Table 13: Correlation data for FOLIO($\mathcal{I}$). The ∀uto∃∨∧L data was averaged from the FOL dataset with data points with description complexity $d \leq 6$. These values were used to compute the predictive power of ∀uto∃∨∧L reported in Fig. 4.

Model		FOLIO($\mathcal{I}$) Score			
Model	∀uto∃∨∧L Score	BLEU	ROUGE	METEOR	BERT
GPT-4o	0.71	0.14	0.42	0.64	0.71
GPT-4o-mini	0.49	0.13	0.41	0.61	0.73
ChatGPT	0.27	0.19	0.47	0.62	0.76
Mistral-7B-Instruct-v0.2	0.04	0.08	0.31	0.51	0.64
Phi-3-Medium-4k-Instruct	0.26	0.12	0.39	0.58	0.70
LLama-3-8B-Instruct	0.08	0.04	0.18	0.35	0.50
Gemma-2-9B-IT	0.21	0.10	0.34	0.53	0.63
Granite-3.0-8B-Instruct	0.12	0.15	0.41	0.58	0.72
Llama-3.1-8B-Instruct	0.06	0.09	0.31	0.51	0.64
LLama-3.2-1B-Instruct	0.02	0.00	0.06	0.15	0.36
LLama-3-70B-Instruct	0.39	0.12	0.40	0.60	0.70
Ministral-8B-Instruct-2410	0.07	0.11	0.36	0.55	0.67
Phi-3.5-Mini-Instruct	0.12	0.05	0.22	0.41	0.55
Qwen-2.5-1.5B-Instruct	0.05	0.09	0.33	0.49	0.65
Qwen-2.5-14B-Instruct	0.57	0.07	0.26	0.45	0.55
Yi-1.5-34B	0.14	0.12	0.39	0.58	0.72

Table 14: Correlation data for HumanEval ($\mathcal{A}$). The $\forall$uto$\exists\vee\wedge$L data was averaged from the regex dataset with data points with description complexity $d \leq 7$. These values were used to compute the predictive power of $\forall$uto$\exists\vee\wedge$L reported in Fig. 4.

Model	$\forall$uto$\exists\vee\wedge$L Score	HumanEval ($\mathcal{A}$) Score
GPT-4o	0.66	0.92
GPT-4o-mini	0.44	0.88
ChatGPT	0.36	0.74
Mistral-7B-Instruct-v0.2	0.20	0.45
Phi-3-medium-4k-insruct	0.45	0.75
LLama-3-8B-Instruct	0.07	0.63
Gemma-2-9B-IT	0.28	0.68
Granite-3.0-8B-Instruct	0.21	0.62
Llama-3.1-8B-Instruct	0.19	0.63
LLama-3.2-1B-Instruct	0.03	0.35
LLama-3-70B-Instruct	0.33	0.80
Ministral-8B-Instruct-2410	0.13	0.77
Phi-3.5-Mini-Instruct	0.36	0.71
Qwen-2.5-1.5B-Instruct	0.12	0.57
Qwen-2.5-14B-Instruct	0.45	0.81
Yi-1.5-34B	0.13	0.73