arXiv:2608.30258v1 [cs.CL] 31 Aug 2026

Stratified Consistency Distillation for Natural Language Formalization

Zhichao Hou[1] **Ferhat Erata**[2] **Joe Lilien**[2] **MohamadAli Torkamani**[2*]

[1]North Carolina State University, [2]Amazon Web Services

zhou4@ncsu.edu, {erata,lilienj,alitor}@amazon.com

Abstract

Neurosymbolic reasoning has shown promising success in addressing complex reasoning tasks by combining large language models (LLMs) and symbolic solvers. While this approach shows promise, a fundamental challenge remains: improving the accuracy of translations from natural language to logical formulas. Current methods predominantly rely on prompt engineering, which is difficult to scale across different domains and input formats. Drawing inspiration from the success of fine-tuning in other model adaptation and alignment applications, we propose a fine-tuning-based Stratified Consistency Distillation approach: (1) We generate K logical translations per input using a frontier LLM and cluster them by semantic equivalence (2) Based on the entropy level, we apply majority voting (low entropy), LLM-as-a-Judge (medium entropy), or unification/abstention (high entropy), and (3) fine-tune a smaller model using the selected pseudo-labels. Our experiments show significant and consistent improvements in both Pass@K and our novel Equivalent Logical Similarity metrics, demonstrating the potential of advancing logical translation through consistency distillation.

1 Introduction

Large Language Models (LLMs) have become a cornerstone technology for building customer-facing chatbot systems, enabling natural, context-aware, and highly interactive conversations across diverse application domains. Despite their impressive capabilities, LLMs are inherently prone to hallucinations—the generation of factually incorrect or logically inconsistent outputs—which may directly contradict authoritative source-of-truth documents. Such errors can be catastrophic in high-stakes domains such as compliance, pricing, and regulations, where a single incorrect response may result in financial loss, legal liability, or reputational damage.

A principled way to ensure the *logical soundness* and *policy compliance* of chatbot responses is to convert human-written natural language into *verifiable formal representations* (e.g., SMT-LIB) and to verify their correctness using automated reasoning tools such as the Z3 solver. This leads us to a fundamental research question in the emerging field of *LLM reasoning and autoformalization*:

> *Can LLMs accurately understand human-written natural language and translate it into formal logical formulas that can be verified by a symbolic solver?*

Recent advances in frontier LLMs (OpenAI & et al., 2024; Anthropic, 2024; Team & et al., 2024)—empowered by few-shot in-context learning and chain-of-thought prompting (Fu et al.,

*Corresponding author.

Preprint.

2023; Wei et al., 2023)—have demonstrated remarkable capabilities in such translation tasks. However, prompt-based approaches using frontier LLMs face two major limitations: (1) Frontier models are extremely large, with parameter counts ranging from 70B to over 500B, leading to high inference latency and prohibitive computational cost; and (2) These models are generally closed-source and only available through black-box APIs, preventing fine-tuning and constraining performance improvements beyond prompt engineering.

To address these limitations, we propose a systematic and scalable framework—*Stratified Consistency Distillation*—for translating policy-governed natural language into formal logic, which distills the reasoning and translation capabilities of a frontier LLM into a smaller, more efficient open-source language model. Our main contributions are summarized as follows:

- **Synthetic Dataset Generation from Policy Documents.** We design a robust pipeline to automatically extract domain-relevant rules from unstructured policy documents and generate aligned natural language–SMT-LIB training pairs, enabling scalable data creation without manual annotation.
- **Stratified Consistency Distillation (SCD).** We introduce a novel distillation strategy that leverages semantic equivalence clustering and entropy-based stratification to selectively transfer knowledge from a high-capability frontier LLM into a smaller LM, preserving logical consistency while reducing inference cost.
- **Comprehensive Evaluation and Analysis.** We conduct extensive experiments across multiple policy-driven reasoning benchmarks. Our method achieves substantial improvements in logical translation accuracy over both prompt-based frontier LLM approaches and fine-tuned baselines, while offering $5\times$–$20\times$ lower inference cost.

2 Preliminary

As large language models (LLMs) are increasingly used to power customer-facing chatbots, ensuring that their outputs remain accurate and aligned with formal company policies has become a critical challenge, especially in high-stakes domains such as compliance, pricing, and regulation. Despite their impressive capabilities, LLMs may generate hallucinated or logically inconsistent responses that contradict authoritative source documents. To improve chatbot reliability, we combine the natural language understanding capabilities of LLMs with formal logical reasoning, ensuring that generated responses are not only fluent but also logically sound and policy-compliant. In this section, we introduce our pipeline for generating synthetic data from policy documents, the resulting SMT-LIB representation, the NL2SMT problem formulation, and the evaluation metrics.

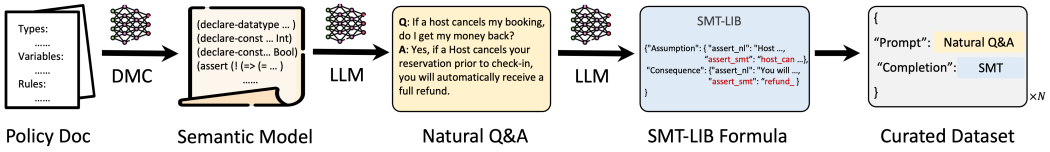

Figure 1: Synthetic dataset generation pipeline.

Synthetic dataset generation from policy documents. We generate natural language question-answer pairs and their corresponding SMT-LIB translations using information extracted from source policy documents. The pipeline consists of the following steps, as illustrated in Figure 1:

1. *Extract semantic context.* The process begins with a source policy document. An LLM is used to construct a semantic model of the document, including an SMT-LIB specification consisting of relevant declarations, variables, and policy rules. The extracted declarations and contextual information guide the subsequent data-generation steps.
2. *Generate natural language Q&A pairs.* An LLM generates natural language question-answer pairs based on the extracted semantic context. These pairs represent realistic user-chatbot interactions and are grounded in the content and rules of the source policy document.
3. *Translate natural language into SMT-LIB.* The generated natural language content and the extracted semantic context are incorporated into the translation prompt. Using its reasoning capabilities, the LLM translates each natural language instance into a complete and syntactically valid SMT-LIB representation that captures the logical semantics of the question-answer pair.

4. *Construct the final training pairs.* Each generated example is organized in the following format:

   `<Natural Language Prompt>` $\rightarrow$ `<SMT-LIB Completion>`.

   These prompt-completion pairs form the training dataset used to align natural language inputs with their formal logical representations.

SMT-LIB data representation. The resulting dataset consists of natural language prompts paired with their corresponding SMT-LIB translations. Each translation expresses the logical content of the input using the declarations, variables, and rules extracted from the relevant semantic context. Depending on the input, an SMT-LIB completion may contain logical operators, quantified expressions, constraints, or relationships between conditions and their consequences. All examples nevertheless follow a unified SMT-LIB representation and are treated as instances of the same NL2SMT translation task.

NL2SMT problem setup. We aim to improve the translation of natural language inputs into formal SMT-LIB representations. Given a natural language prompt $\mathbf{p}$, which may include task instructions, contextual information, and a question-answer pair, the goal is to generate an SMT-LIB completion $\mathbf{t}$ that accurately captures its logical semantics. Formally, we define a training dataset $\mathcal{D}$ consisting of paired samples $(\mathbf{p}, \mathbf{t})$, where $\mathbf{t}$ is the ground-truth SMT-LIB translation of $\mathbf{p}$. Let $\mathbb{LM}_\theta$ denote a parameterized language model with parameters θ. Our objective is to maximize the expected similarity $\mathcal{S}$ between the generated translation $\mathbb{LM}_\theta(\mathbf{p})$ and the ground-truth translation $\mathbf{t}$, where $\mathcal{S}$ measures either exact logical equivalence or continuous logical similarity:

$$\max_{\theta} \mathbb{E}_{(\mathbf{p},\mathbf{t})\sim\mathcal{D}} \left[\mathcal{S}\left(\mathbb{LM}_\theta(\mathbf{p}), \mathbf{t}\right)\right]. \tag{1}$$

Equivalence measurements. To evaluate the quality of the generated SMT-LIB translations, we consider two complementary measurements of $\mathcal{S}$:

- *Binary Equivalence Check.* We use the Z3 theorem prover to determine whether a generated SMT-LIB formula is logically equivalent to its ground-truth counterpart. The check returns `True` when the two formulas are equivalent and `False` otherwise. We report **Pass@10**, which measures whether at least one of ten generated candidates is logically equivalent to the ground-truth formula.
- *Continuous Similarity Score.* When exact equivalence is not achieved, we compute a graded similarity score in the range $[0, 1]$ using structural compression and anti-unification over the generated and ground-truth SMT-LIB formulas. Specifically, we use Egglog to extract an anti-unifier according to the defined SMT-LIB specification. The resulting score measures the degree of shared logical structure between the formulas.

Together, these measurements provide both a strict assessment of logical equivalence and a continuous assessment of partial logical alignment between generated and ground-truth SMT-LIB formulas.

3 Stratified Consistency Distillation

In this section, we introduce a systematic framework for logical translation-Stratified Consistency Distillation-which transfers knowledge from a frontier LLM to a smaller LM. The systematic overview of our framework is provided in Figure 2.

3.1 Stratified Consistency Distillation

The pretrained frontier large language models (LLMs) (OpenAI & et al., 2024; Anthropic, 2024; Team & et al., 2024) have demonstrated superior reasoning capabilities, especially when combined with prompting techniques such as chain-of-thought (Fu et al., 2023; Wei et al., 2023). However, deploying these powerful LLMs in real-world scenarios remains challenging due to two main limitations: (1) They are typically extremely large, with model sizes ranging from 70B to over 500B parameters, resulting in significant latency and computational overhead during inference; and (2) These models are generally closed-source and only accessible through black-box APIs, which prevents fine-tuning and thus limits performance improvements beyond prompt engineering.

To overcome these limitations, we propose a Stratified Consistency Distillation pipeline that transfers knowledge from frontier teacher LLMs to a smaller student generator, as illustrated in Figure 2(a). At a high level, our pipeline consists of four main steps:

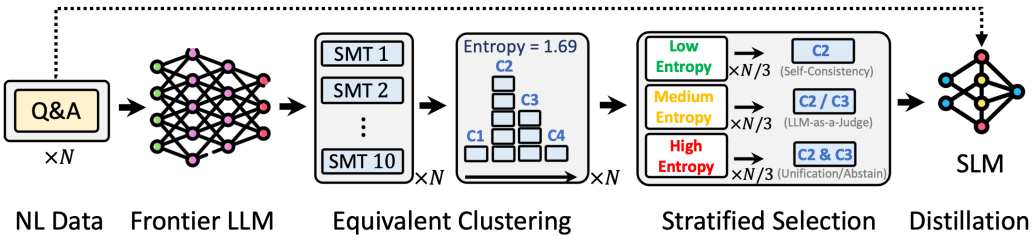

Figure 2: Schematic overview of Stratified Consistency Distillation: For each training sample, we generate 10 SMT-LIB translations using a frontier LLM (e.g., Claude Sonnet 3.7). These translations are clustered based on semantic equivalence, and semantic entropy is computed. The data is then stratified into three groups by entropy: *Low entropy*: Apply majority voting for self-consistency; *Medium entropy*: Use LLM-as-a-Judge to select among the top-2 clusters; *High entropy*: Perform unification of the top-2 translations or abstain the data. Finally, we distill the knowledge into a smaller model (e.g., Qwen) via fine-tuning on Q&A pairs and pseudo-labels derived from stratified selection.

1. **Generation:** Sample output sequences of tokens from the predictive distribution of a frontier LLM given an input prompt $\mathbf{p}$.
2. **Clustering:** Cluster the generated translations based on their logical equivalence using our proposed clustering algorithm. Estimate semantic entropy by summing the probabilities of clusters, following a defined entropy formulation.
3. **Selection:** Select pseudo labels using different strategies depending on the estimated entropy.
4. **Distillation:** Use the input and selected pseudo label from the frontier LLM to train a smaller student model.

Redundant Generation. Given a prompt $\mathbf{p}$ (including necessary instructions, few-shot examples, and the Q&A to be translated), we sample M SMT-LIB translations $\{\mathbf{t}^{(1)}, \mathbf{t}^{(2)}, \ldots, \mathbf{t}^{(M)}\}$ from a frontier LLM (e.g., Claude Sonnet 3.7 (Anthropic, 2024)), denoted as $\mathbb{LLM}$. To accelerate the sampling process, we employ `vLLM` (Kwon et al., 2023), a high-throughput and memory-efficient inference and serving engine.

Equivalent Clustering & Symbolic Semantic Entropy. We invoke our `check_smt_equivalence` function $\mathbb{E}(\cdot,\cdot)$ to group the M translations into equivalence clusters based on logical equivalence. The `check_smt_equivalence` function verifies whether two SMT-LIB expressions (`smt1` and `smt2`) are logically equivalent under given `declarations` using the Z3 SMT solver. The function constructs implication trees, writes them to an SMT-LIB file, and checks the satisfiability of the negated equality condition. A return value of `unsat` from Z3 indicates logical equivalence. The function outputs a dictionary containing the equivalence result, Z3 output, and any errors. Semantic entropy (Farquhar et al., 2024) has been introduced in natural language reasoning as a means to improve the correctness and soundness of LLM-generated reasoning. Extending this concept, we introduce symbolic semantic equivalence as an alternative formulation, described as follows. Recall that an equivalence relation is reflexive, symmetric, and transitive. Any such relation induces a set of equivalence classes. Each semantic equivalence class groups outputs expressing the same logical meaning. That is, for the set of classes $\mathcal{C}$, all sentences $\mathbf{t}, \mathbf{t}' \in \mathbf{c} \in \mathcal{C}$ satisfy $\mathbb{E}(\mathbf{t}, \mathbf{t}') = \texttt{True}$. New sentences are added to an existing class if they match any existing member, otherwise a new class is formed. Given the clusters $\mathcal{C}$, we compute semantic entropy as: $\mathrm{SE}(\mathbf{p}) = -\sum_{i=1}^{|\mathcal{C}|} P(\mathcal{C}_i|\mathbf{p}) \log P(\mathcal{C}_i|\mathbf{p})$, where $P(\mathcal{C}_i|\mathbf{p})$ is the normalized probability of cluster $\mathcal{C}_i$.

Stratified Selection for Pseudo Labels. After computing entropy for each input, we construct a training dataset $\left\{\mathbf{p}_i, \{\mathbf{t}_i^{(j)}\}_{j=1}^{M}, e_i\right\}_{i=1}^{N}$, where e_i is the entropy value. We stratify the data into three groups based on entropy and apply different strategies to select pseudo labels:

- *Low entropy*: The generation distribution is concentrated on a dominant cluster. We select a translation from the largest cluster: $\mathbf{t} \in \arg\max_{\mathcal{C}_i \in \mathcal{C}} |\mathcal{C}_i|$.
- *Medium entropy*: There is some ambiguity across top candidates. We use the frontier LLM as a judge to select from the top-2 clusters.
- *High entropy*: High disagreement across generations necessitates either unifying top translations or abstaining from using the sample.

Knowledge Distillation. Following the pipeline above, we obtain a final dataset $\{(\mathbf{p}_i, \mathbf{t}_i)\}_{i=1}^{N}$. We fine-tune a smaller student model (e.g., `Qwen2.5-7B`) using LoRA, distilling knowledge from the frontier LLM.

4 Experiment

4.1 Experiment Settings

We conduct experiments to evaluate the translation of natural language prompts into SMT-LIB formulas under the NL2SMT framework.

Datasets. We evaluate our method on NL2SMT FOLIO dataset. FOLIO is an open-domain first-order logic reasoning benchmark comprising natural language premises and conclusions paired with SMT-LIB assertions constructed using predefined logical declarations. Together, these datasets evaluate NL2SMT translation in both realistic policy-oriented conversations and controlled logical-reasoning scenarios.

Training Strategy. We fine-tune pretrained language models using *LoRA* (Low-Rank Adaptation), a parameter-efficient fine-tuning method. We use a learning rate of 5×10^{-5}, a batch size of 32, a LoRA rank of 32, and a LoRA scaling factor α of 64.

Evaluated Models. Our primary student model is Qwen2.5-7B-Instruct, on which all fine-tuning experiments and ablation studies are conducted. For comparison, we evaluate the few-shot performance of several open-source and proprietary LLMs, including Qwen2.5-7B-Instruct, Qwen3-4B, Qwen3-8B, Qwen3-14B, Mistral-7B-Instruct, and Claude Sonnet 3.7.

Evaluation Metrics. We adopt two complementary metrics to evaluate translation quality. First, for the *Binary Equivalence Check*, we use the Z3 theorem prover to determine whether a generated SMT-LIB formula is logically equivalent to the ground-truth formula. We report Pass@10, which measures whether at least one of ten generated candidates passes the equivalence check. Second, for the *Continuous Similarity Score*, we measure partial logical similarity in the range $[0, 1]$ when exact equivalence is not achieved. This score is computed through symbolic compression and anti-unification of the generated and ground-truth SMT-LIB formulas using an SMT-LIB specification implemented in Egglog.

4.2 Logical NL2SMT Translation

Table 1: Pass@K scores (%) for FOLIO Translation.

Model	Pass@10	Pass@9	Pass@8	Pass@7	Pass@6	Pass@5	Pass@4	Pass@3	Pass@2	Pass@1
Qwen3-4B	19.792	19.792	19.792	19.792	19.792	19.792	19.792	19.792	18.750	17.708
Qwen3-8B	18.750	18.750	18.750	18.750	18.750	18.750	16.667	16.667	16.667	16.667
Qwen3-14B	42.708	42.708	42.708	42.708	42.708	42.708	42.708	42.708	42.708	42.708
Mistral-7B-Instruct	6.250	6.250	6.250	6.250	6.250	5.208	5.208	4.167	4.167	4.167
Qwen2.5-7B-Instruct	21.875	21.875	21.875	21.875	20.833	19.792	18.750	18.750	18.750	15.625
Distillation	50.347	50.347	49.306	48.264	47.222	46.181	45.486	43.750	42.708	39.931
SCD	55.208	55.208	54.514	52.778	50.083	49.653	48.958	47.917	46.528	44.097

FOLIO Dataset. We evaluate our method on FOLIO and report Pass@K for $K = 1, \ldots, 10$ in Table 1. We make the following observations:

- Both distillation methods substantially outperform the pretrained baselines. For example, Qwen2.5-7B-Instruct achieves a Pass@10 of 21.875% in the few-shot setting, whereas vanilla distillation improves it to 50.347%.
- Vanilla distillation also outperforms the strongest pretrained baseline, Qwen3-14B, by 7.639 percentage points in Pass@10 (50.347% versus 42.708%), despite using a smaller student model.
- SCD achieves the best Pass@10 performance of 55.208%, exceeding Qwen3-14B by 12.500 percentage points and vanilla distillation by 4.861 percentage points.

4.3 Downstream Analysis

Visualization. Figures 3 and 4 visualize the per-example Pass@10 outcomes and continuous similarity scores, respectively, under different model and training configurations. In Figure 3,

red denotes an unsuccessful translation and green denotes a successful translation. In Figure 4, colors range from red for low similarity to green for high similarity. The evaluated configurations include pretrained Mistral-7B-Instruct, pretrained Qwen2.5-7B-Instruct, Qwen2.5-7B-Instruct fine-tuned on NL2SMT dataset, and the corresponding model trained using our consistency-distillation framework. The Pass@10 heatmaps show a gradual transition from predominantly unsuccessful predictions for the pretrained models to broader coverage of correct translations after fine-tuning and distillation. Mistral-7B-Instruct exhibits the sparsest coverage of successful predictions, followed by pretrained Qwen2.5-7B-Instruct. Fine-tuning on the NL2SMT dataset increases the number of correctly translated examples, while consistency distillation produces the broadest coverage. The similarity heatmaps exhibit a comparable trend. The pretrained models contain larger regions of low similarity, whereas fine-tuning and consistency distillation shift the distribution toward higher similarity values, indicating stronger logical alignment with the ground-truth translations.

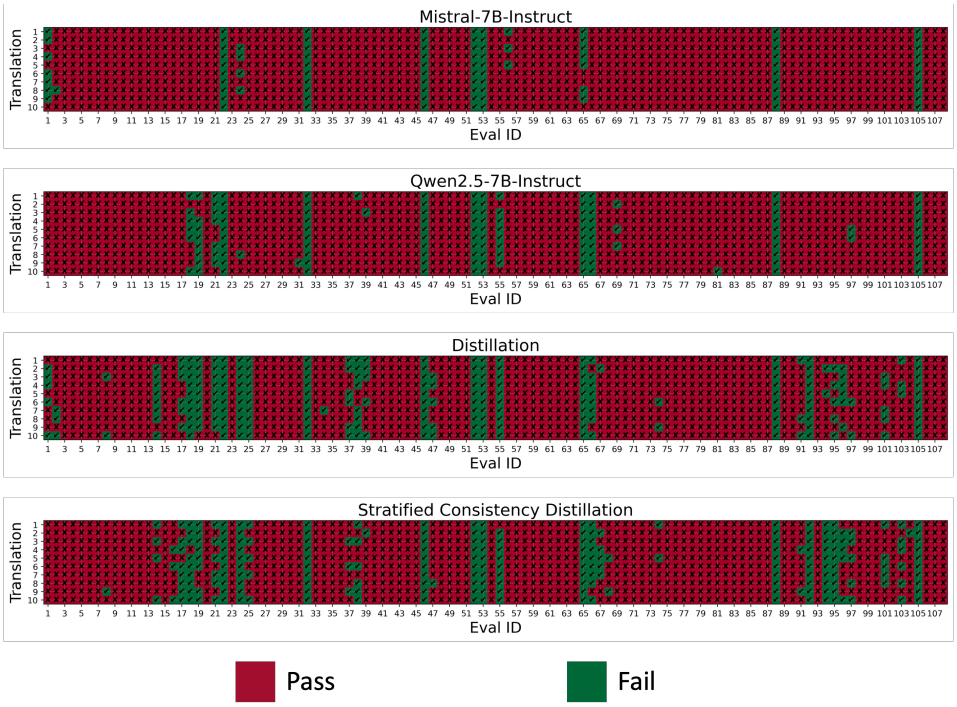

Figure 3: Per-example Pass@10 results under different model and training configurations.

Latency. Table 2 reports the P50, P90, and P99 inference latency of the evaluated models. The fine-tuned Qwen2.5-7B-Instruct model achieves substantially lower latency than Claude Sonnet 3.7 across all three percentiles. Its P50 latency is 4.040 seconds, approximately $4.1\times$ faster than the 16.680 seconds required by Claude Sonnet 3.7. The same advantage is observed at P90 (5.074 versus 28.042 seconds) and P99 (5.651 versus 29.425 seconds), indicating that the efficiency improvement remains consistent at higher latency percentiles. Among the open-source models, Qwen2.5-7B-Instruct also provides a favorable efficiency profile. It achieves lower P90 and P99 latency than Qwen3-4B, Qwen3-14B, and Mistral-7B-Instruct, although Qwen3-8B is faster. These results demonstrate that our approach improves translation quality while retaining practical efficiency.

Table 2: Latency percentiles (P50, P90, P99) of different models on P4d EC2 instance.

Model	**P50**	**P90**	**P99**
Claude Sonnet 3.7	16.680	28.042	29.425
Qwen2.5-7B-Instruct	4.040	5.074	5.651
Qwen3-4B	4.294	5.698	6.281
Qwen3-8B	2.860	4.809	4.835
Qwen3-14B	7.454	13.939	14.604
Mistral-7B-Instruct-v0.2	4.350	7.307	14.191

Reliability of Symbolic Semantic Entropy. To validate the motivation for stratified consistency distillation, we investigate whether the sizes of semantic-equivalence clusters provide a reliable signal for selecting the correct translation. We treat *symbolic semantic entropy* as an indicator of prediction uncertainty, as illustrated in Figure 5. In the low-entropy regime, the largest cluster is dominant and is therefore likely to contain the correct translation. In the medium-entropy regime, the correct translation may instead occur in a secondary cluster. In the high-entropy regime, candidate

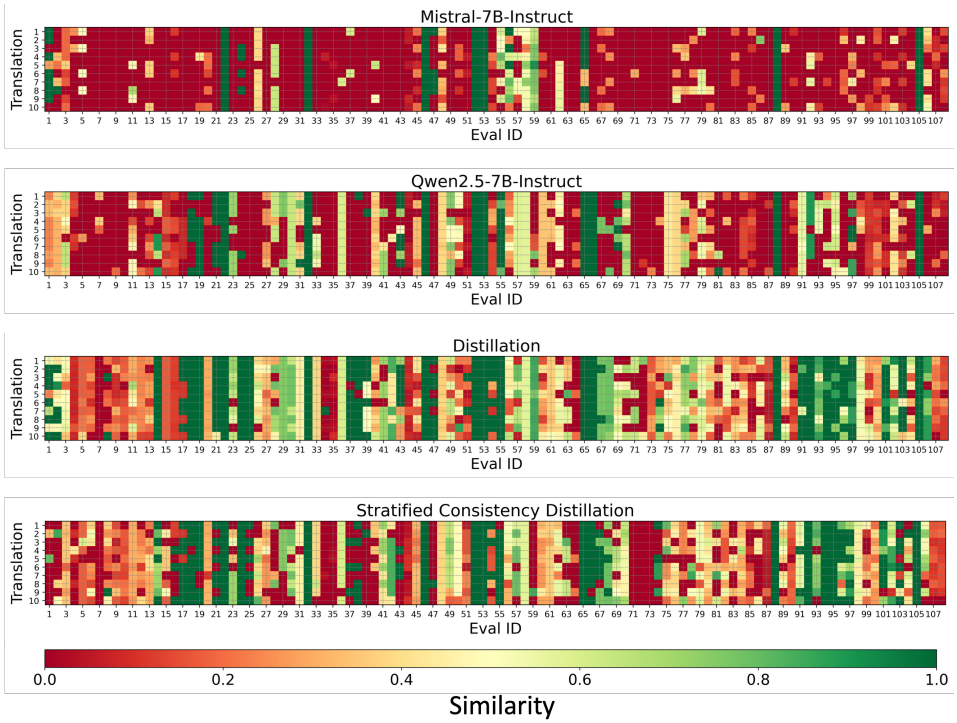

Figure 4: Per-example similarity scores under different model and training configurations.

translations are more evenly distributed across clusters, making the largest cluster less reliable. Table 3 compares different cluster-based selection strategies. Selecting the largest cluster (**Top@1**) performs better than randomly selecting a candidate (**Pass@1**). Moreover, **Top@2** approaches the oracle upper bound represented by **Pass@10**, indicating that the correct translation is usually contained within one of the two largest clusters.

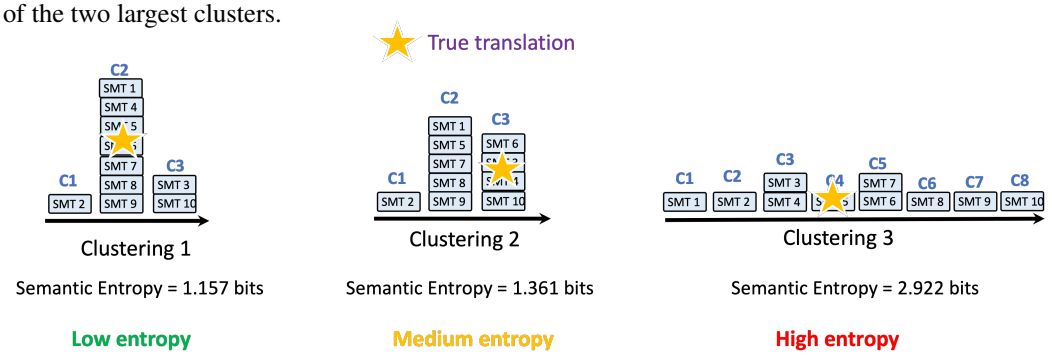

Figure 5: Illustration of semantic-equivalence clusters at different uncertainty levels. Each cluster groups logically equivalent SMT-LIB candidates, and the gold star ($\star$) denotes the ground-truth translation. As entropy increases, the candidates become more evenly distributed across clusters, reducing the reliability of the largest cluster. (a) **Low entropy**: the largest cluster contains the ground truth; (b) **Medium entropy**: the ground truth appears in a secondary cluster; and (c) **High entropy**: the largest cluster is no longer a reliable indicator of correctness.

Table 3: Performance of cluster-based candidate-selection. **Pass@1 (Random)** randomly selects one candidate, whereas **Top@k** considers candidates from the k largest semantic-equivalence clusters.

Model	Pass@10	Pass@1 (Random)	Top@1	Top@2	Top@3
Qwen2.5-7B	17.593	13.889	15.741	17.593	17.593
SCD	31.481	24.074	25.926	30.556	31.481

Stratified Consistency Distillation. The ablation results in Table 4 show that the distillation strategies substantially improve Pass@K over the pretrained Qwen2.5-7B baseline. Vanilla distillation already

produces a considerable improvement, demonstrating the value of transferring high-quality outputs from the teacher model. Among the entropy-specific variants, training with low-entropy samples consistently outperforms training with high-entropy samples across all values of K, suggesting that more consistent teacher predictions provide a stronger supervision signal. Most importantly, stratified SCD achieves the highest or tied-highest performance across all reported Pass@K metrics. This result demonstrates that applying different selection strategies according to semantic entropy produces a more effective and robust distillation signal than relying on a single entropy regime.

Table 4: Ablation study of stratified consistency distillation on the Customer Service dataset.

Model	**Pass@10**	**Pass@9**	**Pass@8**	**Pass@7**	**Pass@6**	**Pass@5**	**Pass@4**	**Pass@3**	**Pass@2**	**Pass@1**
Qwen2.5-7B	17.593	17.593	17.593	17.593	16.667	16.667	15.741	15.741	14.815	13.889
Vanilla Distillation	27.469	27.469	27.116	26.543	25.926	25.617	24.383	23.148	22.222	19.753
SCD (High Entropy)	26.235	26.235	25.000	24.691	23.765	22.679	20.679	19.753	18.519	18.519
SCD (Low Entropy)	28.704	28.704	28.704	28.086	27.778	26.852	25.926	25.926	24.074	23.457
SCD (Stratified)	31.481	31.481	31.481	30.247	29.938	26.852	26.852	26.852	26.852	26.852

5 Related Works

Improving Logical Reasoning of LLMs. Substantial research has been devoted to enhancing the logical reasoning capabilities of large language models (LLMs), with a particular focus on mathematical reasoning tasks. Early studies have shown that pretrained LLMs (Team & et al., 2024; OpenAI & et al., 2024; Anthropic, 2024) can solve reasoning problems through prompting strategies such as Chain-of-Thought (CoT) reasoning (Fu et al., 2022; Wei et al., 2022), which guides the model to generate intermediate steps before producing the final answer. Beyond prompting, supervised fine-tuning (SFT) (Cobbe et al., 2021; Yu et al., 2023) on high-quality, human-annotated datasets has been shown to yield further improvements in reasoning accuracy. More recently, reinforcement learning from human feedback (RLHF) (Ziegler et al., 2019) has emerged as a powerful approach for improving reasoning performance, leveraging reward models (Lightman et al., 2023; Wang et al., 2023) to align LLM outputs with desired reasoning processes and solutions. However, these techniques for translating informal natural language into formal SMT-LIB formulas remain largely underexplored.

Automalization of LLMs. Autoformalization, the task of converting informal natural language into verifiable formal representations, plays a foundational role in both mathematical formalization and the emerging verification of LLM-generated outputs. In mathematical contexts, it enables the translation of human-written proofs into machine-checkable formats for proof assistants such as Coq (Team, 2024), Lean (De Moura et al., 2015), and Isabelle (Ait Mohamed et al., 2008). More recently, its scope has expanded to address reliability issues in LLMs by translating generated text into precise, logically consistent forms using systems such as first-order logic (Ryu et al., 2024) or arithmetic frameworks like Peano arithmetic (Kennedy & Amsler, 1974). By bridging the expressive flexibility of natural language and the rigor of formal verification, autoformalization mitigates semantic ambiguity and supports robust reasoning. However, translating LLM outputs directly into SMT-LIB—a critical formalism for automated reasoning over logical constraints—remains largely unexplored. This gap motivates our work, which aims to advance LLM automatization in this under-addressed direction.

6 Conclusion

In this work, we addressed the challenge of ensuring the logical soundness of LLM-generated responses in high-stakes domains. We proposed a principled NL2SMT framework that translates natural language into verifiable SMT-LIB formulas, enabling automated verification with symbolic solvers. Our Stratified Consistency Distillation method selectively distills the logical translation capability of a frontier teacher model into a smaller open-source student, allocating supervision according to the uncertainty of the generated translations. Experiments on FOLIO show that our method improves translation accuracy over both pretrained and vanilla-distillation baselines, and matches or exceeds substantially larger models despite using far fewer parameters. The distilled model further offers considerably lower inference latency, making the framework more practical to deploy. Overall, this work provides a scalable route to efficient, reliable, and verifiable language models for domains in which logical correctness is essential.

References

Ait Mohamed, O., Munoz, C., and Tahar, S. *Theorem Proving in Higher Order Logics: 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008, Proceedings*, volume 5170. Springer Science & Business Media, 2008.

Anthropic. Claude 3 model family, 2024. URL https://www.anthropic.com/news/claude-3-family. Accessed: 2024-10-XX.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

De Moura, L., Kong, S., Avigad, J., Van Doorn, F., and von Raumer, J. The lean theorem prover (system description). In *International Conference on Automated Deduction*, pp. 378–388. Springer, 2015.

Farquhar, S., Kossen, J., Kuhn, L., and Gal, Y. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Fu, Y., Peng, H., Sabharwal, A., Clark, P., and Khot, T. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*, 2022.

Fu, Y., Peng, H., Sabharwal, A., Clark, P., and Khot, T. Complexity-based prompting for multi-step reasoning, 2023. URL https://arxiv.org/abs/2210.00720.

Kennedy, H. C. and Amsler, R. *Giuseppe Peano*. Birkhäuser, 1974.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

OpenAI and et al., J. A. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Ryu, H., Kim, G., Lee, H. S., and Yang, E. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. *arXiv preprint arXiv:2410.08047*, 2024.

Team, G. and et al., P. G. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

Team, T. C. D. The coq proof assistant, June 2024. URL https://doi.org/10.5281/zenodo.11551307.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Prompt for Translation Generation

Prompt for Translation Generation (FOLIO)

```
You are an expert in autoformalization - translating natural language
    logical reasoning into SMT-LIB format.

Your task: Given natural language premises and conclusion, generate
    SMT-LIB assertions using the provided declarations.

<instructions>
1. Analyze the SMT-LIB variables provided within the 'declarations'
    section to understand all allowed variables, their types, and
    descriptions.
2. Translate the natural language premises and conclusion into SMT-LIB
    assertions
3. Use only the constants and functions declared in the declarations
    section
4. Each assertion should be wrapped in (assert ...)
5. Use proper SMT-LIB syntax for logical operators: and, or, not, =>,
    xor, forall, exists
6. DO NOT include any XML tags or markdown formatting - output only
    pure SMT-LIB assertions
</instructions>

Here is an example:

<example>
<input>
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 aware that caffeine is a drug. Rina is
    either a student or unaware that caffeine is a drug, but not both.
    Rina is either dependent on caffeine or a student, but not both.
Conclusion: Rina doesn't want to be addicted to caffeine or is unaware
    that caffeine is a drug.
</input>

<declarations>
(declare-sort Individual)
(declare-const rina Individual)
(declare-const coffee Individual)
(declare-const caffeine Individual)
(declare-fun DrinkRegularly (Individual Individual) Bool)
(declare-fun IsDependentOn (Individual Individual) Bool)
(declare-fun WantToBeAddictedTo (Individual Individual) Bool)
(declare-fun AwareThatDrug (Individual Individual) Bool)
(declare-fun Student (Individual) Bool)
</declarations>

Output (pure SMT-LIB assertions only):
(assert (forall ((x Individual)) (=> (DrinkRegularly x coffee) (
    IsDependentOn x caffeine))))
(assert (forall ((x Individual)) (or (DrinkRegularly x coffee) (not (
    WantToBeAddictedTo x caffeine)))))
(assert (forall ((x Individual)) (=> (not (WantToBeAddictedTo x
    caffeine)) (not (AwareThatDrug x caffeine)))))
(assert (not (xor (Student rina) (not (AwareThatDrug rina caffeine)))))
(assert (not (xor (IsDependentOn rina caffeine) (Student rina))))
(assert (or (not (WantToBeAddictedTo rina caffeine)) (not (
    AwareThatDrug rina caffeine))))
</example>
```

```
Now translate this natural language input using the provided
    declarations:

<input>
Premises: No sandwich cookies are healthy.
Oreos are sandwich cookies.
Conclusion: All sandwich cookies are delicious.
</input>

<declarations>
(declare-sort Individual)
(declare-const oreos Individual)
(declare-fun SandwichCookie (Individual) Bool)
(declare-fun Healthy (Individual) Bool)
(declare-fun Delicious (Individual) Bool)
</declarations>
```