DISTILLING SMT SOLVER REASONING INTO COMPACT LANGUAGE MODELS

Emre Kıyak* Koç University
Çağatay Cingöz* Koç University
Hakan Çapuk Koç University
Aykut Erdem Koç University

ABSTRACT

We conduct an extensive and comparative analysis of compact language models with fewer than 10B parameters in the domain of Satisfiability Modulo Theories (SMT). While recent works have shown that billion-parameter models can perform logical reasoning, their application to formal SMT solvers like Z3 De Moura & Bjørner (2008) remains underexplored, particularly regarding the trade-off between translation (parsing Natural Language to SMT) and execution (imitating solver reasoning traces). Additionally, while state-of-the-art commercial LLMs can perform strongly on SMT-type problems, the accuracies of smaller models on such tasks remain particularly low due to their limited capacity in arithmetic reasoning tasks. We explore whether it is possible to create a new lightweight language model that can perform well on SMT-based arithmetic and boolean reasoning tasks. Our contribution is a rigorous evaluation of whether resource-constrained models can serve as reliable interfaces for formal verification tools, either by approximating the internal reasoning steps of a solver and accurately predicting the satisfiability of the problem without calling external tools or by correctly formalizing natural language constraints and parsing them correctly in an SMT problem format that a solver like Z3 can solve. Additionally, we evaluate how supervised solver guided training transfers to a related real-world analytical reasoning task in order to assess the limitations of our approach.

1 INTRODUCTION

Despite rapid progress in large language models, reliable reasoning over formal constraints remains a fundamental challenge. Symbolic systems such as Satisfiability Modulo Theories (SMT) solvers can verify arithmetic and logical statements with mathematical precision, but are limited to structured inputs and cannot interpret natural language. In contrast, modern language models excel at processing and generating natural language, yet they are fundamentally probabilistic and often fail on tasks requiring precise symbolic or arithmetic reasoning, especially without access to external tools.

This contrast highlights a fundamental gap between symbolic reasoning systems, which are exact but rigid, and neural language models, which are flexible but unreliable for formal reasoning. While recent large-scale models such as GPT-5 and Gemini-3 demonstrate strong performance on SMT-style tasks, these capabilities rely on massive scale and complex architectures. Compact open-source models, in contrast, still struggle with such tasks. Understanding whether this gap can be closed or whether it reflects inherent limitations of compact models is both scientifically and practically important. SMT problems provide a controlled and structured testbed for evaluating reasoning capabilities in language models. Solving SMT constraints requires consistent symbolic manipulation, arithmetic reasoning, and logical inference, making it difficult to succeed through superficial pattern matching alone.

In this work, we investigate whether compact, resource-constrained language models (2B–8B parameters) can learn to approximate the internal reasoning behavior of an SMT solver. We explore two complementary learning paradigms:

*Equal contribution

1. **Solver-trace imitation:** training models to reproduce structured reasoning steps and predict satisfiability outcomes without external tool access.
2. **Natural language to SMT translation:** training models to convert natural language descriptions of constraints into formal SMT representations that can be solved symbolically.

These two perspectives capture complementary aspects of reasoning required for SMT tasks. Solver-trace imitation evaluates whether models can internalize structured symbolic inference and maintain arithmetic consistency, while natural language to SMT translation measures the ability to map unstructured language into formal representations. The latter can also be combined with external solvers, enabling lightweight neuro-symbolic pipelines without requiring large-scale models.

To maintain computational efficiency, we adopt parameter-efficient fine-tuning using Low-Rank Adaptation (LoRA) Hu et al. (2022), avoiding full model retraining. This allows us to study reasoning capabilities in compact models under realistic resource constraints while preserving their pretrained knowledge.

Our goal is not only to improve performance on SMT-style tasks, but also to better understand the extent to which structured symbolic supervision can shape reasoning behavior in small language models. In particular, we investigate whether such supervision transfers beyond synthetic tasks to related analytical reasoning settings.

In summary, our contributions are:

- We propose a framework for distilling SMT solver reasoning into compact language models, enabling them to approximate symbolic reasoning without external tool calls.
- We introduce two complementary training paradigms—solver-trace imitation and natural language to SMT translation—that jointly capture internal reasoning dynamics and semantic parsing capabilities.
- We provide empirical evidence that parameter-efficient, solver-guided supervision improves performance on SMT tasks and transfers to out-of-distribution analytical reasoning benchmarks.

2 Related Work

LogicLM Pan et al. (2023) uses LLMs to generate symbolic formulations that are executed via a diverse set of external solvers such as Pyke (Deductive Reasoning), Prover9 (First-Order Logic), python-constraint (Constraint Satisfaction) and Z3 (Analytical Reasoning). LogicLM++ Kirtania et al. (2024) extends LogicLM by optimizing natural language to symbolic formulation parsing through feeding a pairwise comparison prompt to LLM for obtaining the most reliable formulation. In contrast to LogicLM, we attempt to develop language models that are capable of doing logical reasoning without necessarily relying on tools. This study also utilizes state-of-the-art LLMs of its time, such as GPT-4 and GPT-3.5 to generate symbolic formulations instead of fine-tuning a lightweight open-source LLM for its specific purpose.

LoGiPT Feng et al. (2023) uses LLMs to generate symbolic forms of natural language logical questions. The authors finetune language models with pairs of symbolic solver reasoning traces and answers so that the models can imitate behavior of such solvers. In particular, they focus on deductive reasoning problems, which are among the most suitable problem types for language models to learn. Whereas, we work on analytical reasoning problems which is underexplored and harder for an LLM to learn.

Program of Thoughts Prompting Chen et al. (2022) and PAL Gao et al. (2023) teach LLMs to generate executable Python programs for numeric reasoning tasks. However, these focus on pure arithmetic computation rather than constraint satisfaction and satisfiability like SMT solvers.

Recent compact models (Phi-3, Gemma-2B, Qwen-2.5) show strong performance under LoRA fine-tuning. Yet little work explores whether such small models can approximate the structured decision processes of theorem provers.

To the best of our knowledge, no prior work has attempted to transfer SMT solver reasoning (Z3) into LLMs. Also, we haven't come across any prior work that addressed the problem of parsing natural

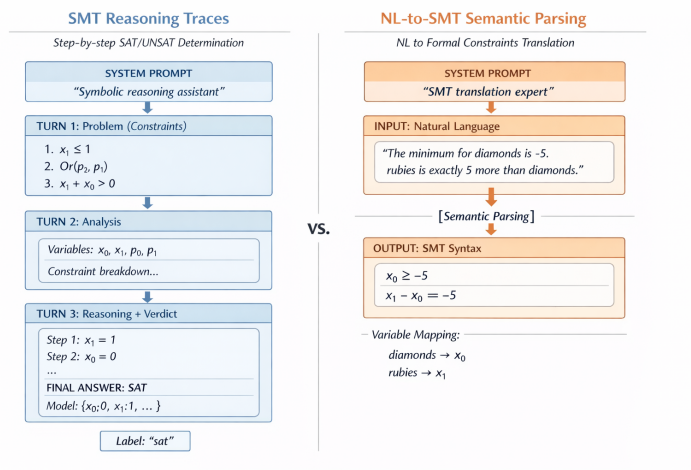

Figure 1: A figure illustrating simplified versions of data samples from both synthetic datasets

language problems to SMT Problem notation. Our work aims to fill these gaps and investigates the scalability limits of solver-trace distillation for compact models.

3 METHODOLOGY

We introduce a multi-stage pipeline approach to evaluate compact models on various SMT tasks. We extend the instruction-tuning methodology of prior neuro symbolic works by introducing a novel dataset generation pipeline specifically for Z3. In addition to synthetic data, we incorporate datasets from prior work. This enables a multi-stage evaluation of compact language models on SMT-related tasks.

3.1 DATA GENERATION PIPELINE

Our data generation pipeline produces two complementary synthetic constraint types built on the Z3 theorem prover: one focusing on arithmetic constraints and the other on boolean constraints. Both share a common constraint generation foundation, with task-specific processing applied downstream.

Core SMT Problem Generation We developed a programmatic constraint generator producing random SMT problems involving linear integer arithmetic and propositional logic. Each problem consists of integer variables $x_0, x_1, \ldots, x_n$ and boolean variables $p_0, p_1, \ldots, p_m$, subject to multiple randomly sampled constraints:

- **Integer inequalities:** $x \geq k$, $x \leq k$, where $k \in [-5, 5]$
- **Algebraic variable arithmetic:** $x + y > k$ and $x - y = k$
- **Boolean constraints:** p, $\neg p$, $p \wedge q$, $p \vee q$, $p \Rightarrow q$

To ensure dataset quality, we apply several filtering mechanisms: eliminating trivial constraints that simplify to tautologies or contradictions using Z3's built-in simplification, performing semantic deduplication by testing whether $c_1 \oplus c_2$ is unsatisfiable (indicating logical equivalence), and bounding all integer variables to $[-20, 20]$.

3.2 DATASET 1: SMT REASONING TRACES

This dataset trains models to determine satisfiability through explicit step-by-step reasoning. For each problem, we invoke Z3 to obtain either a satisfying model or an unsatisfiable core, then construct a structured three-turn conversation:

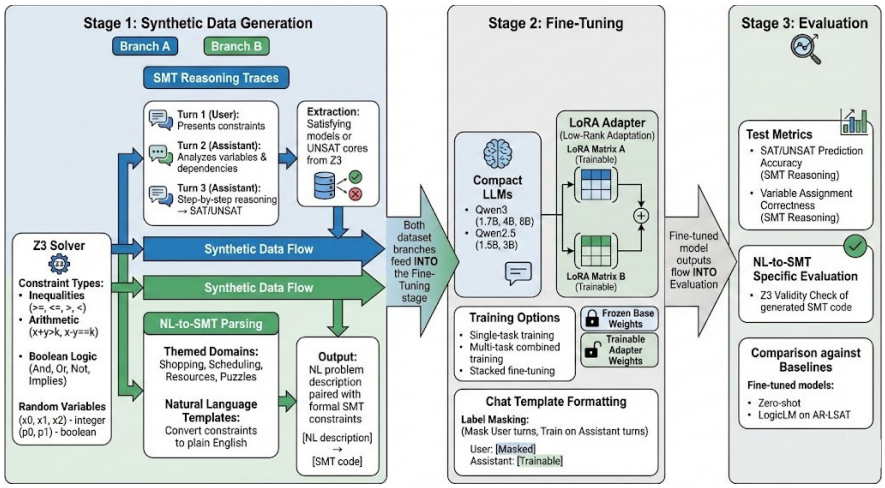

Figure 2: An overview of our end-to-end pipeline and methodology.

Turn 1 (User): Presents the constraint system in symbolic notation with numbered constraints.

Turn 2 (Assistant): Provides structured analysis: variable identification with dependency status (independent, dependent via equality, or jointly constrained), and constraint simplification where beneficial (e.g., $x + x > k$ rewritten as $2x > k$, yielding the integer bound $x \geq \lfloor k/2 \rfloor + 1$).

Turn 3 (Assistant): Contains the complete reasoning trace. For *satisfiable* instances, constraints are processed in priority order (boolean assignments, then equalities, then inequalities, then implications), showing how values are derived rather than merely verified. Logical inference rules (Modus Ponens, Modus Tollens) are applied for implications. A verification section confirms each constraint is satisfied. For *unsatisfiable* instances, we extract the UNSAT core from Z3, analyze each conflicting constraint, and explicitly demonstrate the contradiction, whether from incompatible bounds, boolean conflicts, or equality violations.

We generate balanced datasets of UNSAT examples using rejection sampling.

3.3 Dataset 2: Natural Language to SMT Parsing

This dataset trains models on semantic parsing/translating natural language constraint descriptions into formal SMT notation without determining satisfiability.

Themed Domain Generation. We employ template-based generation with several thematic domains. Each theme provides contextual variable names, boolean condition descriptions, and introductory sentences.

Template-Based Paraphrasing. Each constraint type maps to 6–7 paraphrase templates. For example, $x \geq k$ may render as "The {var} must be at least {k}," "There must be no fewer than {k} {var}," or "At minimum, {var} should be {k}." Similarly, implications use templates like "If {c1}, then {c2}" or "Whenever {c1}, it follows that {c2}." This diversity produces linguistically varied examples from identical underlying constraints.

NL-Optimized Generation. The generator enforces distinct variables in multi-variable constraints ($x \neq y$ in $x + y > k$), preventing awkward constructions like "the apples and the apples combined."

Canonical Variable Renumbering. Variables are indexed based on their order of first appearance in the generated text, not their internal generation order. This ensures consistent mapping between natural language and formal output that models can learn to predict.

Quality Validation. We reject problems with direct boolean contradictions (p and $\neg p$ as separate constraints), require at least one variable to appear in multiple constraints, and filter isolated trivial boolean literals.

The output pairs each natural language description with its formal SMT constraints, stored in JSONL format.

3.4 Fine-Tuning Compact Language Models with LoRA

To adapt compact language models to SMT-related reasoning tasks under limited computational resources, we extend the standard instruction-tuning paradigm by employing Low-Rank Adaptation (LoRA) Hu et al. (2022). Rather than performing full-parameter fine-tuning, which is quite expensive even for models in the 2B–8B parameter range, LoRA enables efficient task adaptation by optimizing only a small subset of parameters while keeping the base model weights frozen.

Let $W \in \mathbb{R}^{d\times k}$ denote a pretrained weight matrix of a transformer layer. In Low-Rank Adaptation (LoRA), the weight update is constrained to a low-rank decomposition:

$$\Delta W = BA, \tag{1}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ with rank $r \ll \min(d, k)$. During fine-tuning, the original weights W are frozen, and only the low-rank matrices A and B are optimized, yielding the effective weight:

$$h = Wx + \Delta W x = Wx + BAx. \tag{2}$$

During backpropagation, gradients are computed only for A and B, significantly reducing the memory overhead for optimizer states and gradient storage.

This formulation drastically reduces GPU memory usage by eliminating gradient storage for the base model and lowering parameter precision, allowing us to fine-tune models with billions of parameters on a single GPU. Importantly, LoRA preserves the expressive capacity of the pretrained model while enabling task-specific adaptation through symbolic supervision.

We apply LoRA-based instruction tuning to several compact open-source models, which include models from Qwen-2.5, and Qwen-3 series, to see how our approach performs on predominantly instruction tuned vs. predominantly reasoning tuned models. All models are trained using identical datasets and supervision formats to ensure a controlled comparison. This setup allows us to track the impact of model scale and symbolic trace supervision on SMT reasoning performance, while remaining within practical computational constraints.

In our implementation, we apply LoRA adapters to all attention and feed-forward projection matrices in the transformer architecture, specifically targeting `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj` modules. We use rank $r = 64$ with scaling factor $\alpha = 128$ for models up to 4B parameters, and $r = 32$, $\alpha = 64$ for larger models to accommodate memory constraints. Training data is formatted using each model's chat template, with label masking applied such that only the assistant's response tokens contribute to the loss computation, system prompts and user turns are masked with label -100 and excluded from gradient updates. We explore three training configurations: (1) single-task training on SMT reasoning traces alone, (2) single-task training on NL-to-SMT parsing alone, and (3) multi-task training where both datasets are concatenated and shuffled, allowing the model to jointly learn constraint reasoning and semantic parsing.

4 Experimental Results

4.1 Evaluation

We evaluate our models across multiple datasets and baselines to assess their performance on SMT-related reasoning and parsing tasks. Our evaluation focuses on both the effectiveness and the benefits of fine-tuning compact language models under limited computational resources.

Datasets. We conduct experiments on three datasets corresponding to different reasoning tasks. *(1) SMT Reasoning Traces:* A synthetically generated dataset consisting of 5,000 samples, where

each instance contains a set of SMT constraints paired with solver-generated reasoning traces and satisfiability labels. This dataset is designed to train and evaluate SAT/UNSAT classification and constraint reasoning. *(2) Natural Language to SMT Parsing:* A synthetic dataset of 5,000 samples containing short natural-language story problems paired with their formal SMT representations. This dataset evaluates the model's ability to translate natural language constraints into executable SMT formulations. *(3) AR-LSAT:* A public benchmark of analytical reasoning questions from the Law School Admission Test. Although it is in the form of natural language multiple choice questions, the structure of the problems show close similarities to our synthetic SMT samples and constraint satisfaction problems as shown by LogicLM Pan et al. (2023). We use AR-LSAT without SMT annotations to test whether reasoning capabilities learned from synthetic SMT supervision transfer to real-world analytical reasoning tasks that involve components resembling formal SMT problems. AR-LSAT is used exclusively for testing purposes and is not included in the training set of the models.

Baselines. We compare our LoRA fine-tuned models against their corresponding base models without task-specific fine-tuning. Base models are evaluated using one-shot prompting, while fine-tuned models are evaluated in a zero-shot setting to fully capture the effect of our SMT training. We additionally report results from prior work on AR-LSAT, particularly LogicLM, as a reference point for neuro-symbolic reasoning performance. We additionally compare the results of LoRA fine-tuned models to their corresponding base models on the AR-LSAT dataset.

Evaluation Metrics. For the SMT reasoning task, we measure overall SAT/UNSAT classification accuracy, per-class accuracy for satisfiable and unsatisfiable instances, and the rate of invalid or missing responses. For the natural language to SMT parsing task, we evaluate exact match accuracy, constraint-level precision, recall, and F1 score, Jaccard similarity between predicted and gold constraint sets, and syntactic validity based on solver executability. For AR-LSAT, we report multiple-choice question accuracy to assess out-of-distribution analytical reasoning performance.

4.2 RESULTS

Our LoRA fine-tuned models exhibit clear and consistent improvements over their base counterparts on both tasks: SMT reasoning (SAT/UNSAT classification) and natural language to SMT semantic parsing. For a fairer comparison, fine-tuned models are evaluated using zero-shot prompts, whereas base models are provided with one-shot prompts that include an example input-expected output pair for the given task.

SMT Reasoning Task. On the SMT reasoning benchmark, fine-tuning leads to substantial accuracy improvements across all model sizes and series. In particular, **Qwen3-8B** shows a drastic gain in overall SAT/UNSAT classification accuracy after LoRA adaptation. As shown in Table 1, while the base **Qwen3-8B** model struggles to reliably solve constraint systems, the fine-tuned variant achieves a markedly higher accuracy, along with improved balance between SAT and UNSAT predictions and a reduced no-answer rate. These results indicate that solver-generated reasoning traces effectively teach the model to internalize structured constraint reasoning rather than relying on surface-level heuristics.

NL-to-SMT Semantic Parsing/Translation Task. Similarly, fine-tuning significantly enhances performance on the natural language to SMT translation task. In recall, F1 and Jaccard similarity metrics all models perform substantially higher post–fine-tuning performance. The fine-tuned model also maintains a **100% syntax validity rate**, demonstrating that improvements stem from better semantic alignment as a result of our guided supervised fine-tuning.

Evaluation on AR-LSAT Test Set Aligned with the efficiency aims of this study, our LoRA fine-tuned and base models are evaluated on AR-LSAT with several constraints: Models are constrained to generate outputs within a limited token budget, with reasoning disabled. In this setting, LoRA fine-tuned **Qwen3-8B** achieves around 20 percent higher absolute accuracy than its corresponding base model. **Qwen2.5-3b** similarly achieves around 10 percent accuracy increase over its base model. LogicLM results from 2023 are shown in table 3 as well, for the same dataset.

Table 1: **Performance on SMT Reasoning Task**

Model	structuredall	SAT	UNSAT	No Ans.
Base Models (One-Shot)				
Qwen3-8B	60.53	44.80	92.00	37.73
Qwen2.5-3B-Instruct	51.07	44.20	64.80	42.93
Qwen2.5-1.5B-Instruct	42.80	44.00	40.40	46.40
LoRA Fine-tuned (Zero-Shot)				
Qwen3-8B	**96.27**	**96.80**	**95.20**	**0.13**
Qwen2.5-3B-Instruct	**94.40**	**95.20**	**92.80**	**0.00**
Qwen2.5-1.5B-Instruct	**94.13**	**95.20**	**92.00**	**0.00**

Table 2: **Performance on NL-to-SMT Parsing Task**

Model	Exact Match	Precision	Recall	F1-Score	Jaccard	Syntax
Base Models (One-Shot)						
Qwen3-1.7B	0.00	0.0286	0.0752	0.0407	0.0257	94.93
Qwen3-4B	0.13	0.4253	0.6621	0.5117	0.3847	97.87
Qwen3-8B	0.13	0.4428	0.6891	0.5392	0.3947	97.07
Qwen2.5-1.5B-Instruct	0.00	0.0258	0.0270	0.0259	0.0165	93.47
Qwen2.5-3B-Instruct	1.60	0.1389	0.2128	0.1599	0.1152	83.07
LoRA Fine-tuned Models (Zero-Shot)						
Qwen3-1.7B	**7.33**	**0.6871**	**0.9998**	**0.8066**	**0.6869**	**100.0**
Qwen3-4B	**14.00**	**0.7054**	**0.9995**	**0.8177**	**0.7053**	**100.0**
Qwen3-8B	**7.47**	**0.6867**	**0.9992**	**0.8061**	**0.6864**	**100.0**
Qwen2.5-1.5B-Instruct	**83.47**	**0.9502**	**0.9979**	**0.9693**	**0.9493**	**100.0**
Qwen2.5-3B-Instruct	**85.47**	**0.9563**	**0.9998**	**0.9737**	**0.9562**	**100.0**

Table 3: **Results of Our Models on AR-LSAT**

Model	Status	Accuracy	Delta vs. Random
Qwen3-8B	Base	7.83%	-12.17%
Qwen3-8B	**LoRA Adapter**	**27.39%**	**+7.39%**
Qwen2.5-3B	Base	13.91%	-6.09%
Qwen2.5-3B	**LoRA Adapter**	**23.48%**	**+3.48%**

LogicLM Pan et al. (2023) Results on AR-LSAT

Dataset	ChatGPT (gpt-3.5-turbo)			GPT-3.5 (text-davinci-003)			GPT-4 (gpt-4)		
	Standard	CoT	Logic-LM	Standard	CoT	Logic-LM	Standard	CoT	Logic-LM
AR-LSAT	20.34	17.31	**26.41**	22.51	22.51	**25.54**	33.33	35.06	**43.04**

4.3 Ablation Studies

Model Size and Series We analyze the effect of model size across SMT reasoning and NL-to-SMT parsing tasks. It appears that the model size only returns marginal improvements on SMT Reasoning and NL-to-SMT parsing tasks. We evaluate three models from the Qwen3 series and two from Qwen2.5. Qwen2.5 series models have undergone an instruction tuning and Qwen3 models are tuned primarily for reasoning in addition to its instruction tuning. On SMT reasoning, models from both families perform similarly overall with a slight edge being on the side of **Qwen3-8B** with 96.27% accuracy. On NL-to-SMT parsing, instruction tuned Qwen2.5 models perform much better, achieving +80% accuracy for exact matching and +95% accuracy for precision, demonstrating the greater success of instruction tuned models in following exact formats.

Combined Dataset Fine-tuning The base models were initially fine-tuned on both our synthetic datasets separately. We choose to evaluate the performance of models that are trained on a combined dataset of SMT Reasoning and NL-to-SMT parsing samples. The base models were trained on 10k sample datasets, 5k from each of our synthetic datasets. No meaningful performance discrepancy emerges between models that were fine-tuned for a single task and models that are fine-tuned with the combination of both datasets, showing that models of even these sizes have the capabilities to specialize in both tasks simultaneously.

AR-LSAT Constraints There are constraints on the models during the AR-LSAT evaluation. The models are restrained in the number of tokens that they should generate while reaching the solution. LoRA fine-tuned **Qwen3-8B** succeeds to reach an answer in less than 128 tokens for each AR-LSAT question reaching an accuracy of 27-29% depending on the seed. Base **Qwen3-8B** achieves 7.83% accuracy in the same setting. When base **Qwen3-8B** is evaluated with 512 tokens per question, it still achieves only 14.34% accuracy. When finally base **Qwen3-8B** is evaluated with unlimited number of tokens per question, it achieves 29.56% accuracy, finally catching up to our fine-tuned **Qwen3-8B** evaluated on less than 128 tokens per question. Our LoRA fine-tuned models achieve comparable results with the base model without the need to produce a large number of inference tokens.

4.4 DISCUSSION

Feasibility of Distilling SMT Reasoning into Compact Models. Our results demonstrate that SMT-related reasoning can be distilled into compact language models. Across multiple model sizes and two different tasks, fine-tuned models consistently outperform their base counterparts under comparable prompting settings, indicating that structured constraint-solving behavior can be transferred through solver-guided supervised training. While our findings are limited to the evaluated tasks and datasets, they provide evidence that compact language models are capable of capturing aspects of SMT-style reasoning patterns.

Generalization to Related Reasoning Tasks. Despite being heavily specialized through fine-tuning on synthetic SMT reasoning and semantic parsing datasets, our models do not exhibit degraded reasoning performance within our experimental scope. Fine-tuned models show improved performance on the out-of-distribution AR-LSAT benchmark, despite never being trained on this dataset. These results show that LoRA-based parameter-efficient fine-tuning enables models to acquire structured symbolic reasoning skills while maintaining performance on related analytical reasoning tasks.

Potential Early Convergence on In-Distribution Tasks. Although the fine-tuned models demonstrate clear performance improvements on in-distribution SMT reasoning and semantic parsing tasks, it is possible that the underlying reasoning mechanisms did not undergo substantial structural change. Given the constrained nature of the synthetic datasets and the relatively well-defined task formats, the models may have converged to effective task-specific solutions early in training without requiring extensive adaptation. This indicates that while solver-guided training improves task accuracy, more complex datasets and reasoning structures may be required to evaluate broader generalization.

Efficient Discovery of Structured Reasoning Capabilities. The fact that our fine-tuned **Qwen3-8B** model substantially outperforms the base **Qwen3-8B** model on AR-LSAT under the same architectural constraints, and even when evaluated under disadvantageous conditions, indicates that our approach introduces a more efficient and structured reasoning process for problems involving constraint satisfiability. Rather than relying on extensive prompting or long reasoning traces at inference time, the model appears to internalize these reasoning patterns during fine-tuning. These results indicate that LoRA-based solver-guided training enables more efficient discovery and utilization of a model's inherent reasoning capabilities.

5 LIMITATIONS AND FUTURE DIRECTIONS

Synthetic Data Bias. A primary limitation of this work is the heavy reliance on synthetically generated SMT problems for training. While synthetic data enables precise supervision and controlled experimentation, it may not fully capture the diversity, ambiguity, and structural irregularities of real-world tasks that build on constraint satisfaction.

Restricted Constraint Expressivity. The SMT problems used in this study focus mainly on linear integer arithmetic and Boolean constraints. More expressive SMT theories, such as non-linear arithmetic, quantifiers, arrays, or bit-vectors, were out of the scope of this study. As a result, the reasoning capabilities learned by the models are limited to a certain subset of SMT problem classes. The fine-tuned models managed to achieve high accuracies in both tasks, which also implies a need for a more expressive dataset with more complexity.

Limited Exposure to Incorrect Reasoning Paths. While our SMT reasoning dataset includes solver-generated step-by-step traces leading to the correct SAT/UNSAT decision, the training samples only expose the model to successful reasoning trajectories. Incorrect or alternative reasoning paths that lead to invalid conclusions are not included in the three-turn conversational structure. Incorporating negative reasoning trees or counterfactual reasoning paths could further strengthen the model's ability to refine its reasoning process.

Reinforcement Learning on Top of LoRA Fine-Tuning. Beyond LoRA-based SFT, reinforcement learning could be applied to fine-tuned models to further align generated reasoning with solver-consistent outcomes. Reward signals derived from Z3 correctness (e.g., satisfiability accuracy or constraint satisfaction) could encourage models to refine their reasoning strategies.

6 CONCLUSION

Our results show that LoRA-based distillation of SMT solver behavior substantially improves both satisfiability prediction and structured semantic parsing in compact language models. The strong performance of **Qwen3-8B** indicates that targeted symbolic supervision can enhance reasoning capabilities even under limited model capacity and training data. Rather than relying on extended prompting or long inference traces, fine-tuned models appear to adopt more structured and efficient reasoning patterns. At the same time, these findings are grounded in synthetic datasets and relatively simple SMT domains, leaving open questions about generalization to more complex and diverse reasoning settings. Evaluating this approach across broader problem classes remains an important direction for future work.

Overall, this work demonstrates that structured symbolic supervision provides a practical and compute-efficient mechanism for improving reasoning in compact models, bringing neural language models closer to the reliability of formal symbolic systems.

REFERENCES

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340. Springer, 2008.

Jiazhan Feng, Ruochen Xu, Junheng Hao, Hiteshi Sharma, Yelong Shen, Dongyan Zhao, and Weizhu Chen. Language models can be logical solvers. *arXiv preprint arXiv:2311.06158*, 2023.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Shashank Kirtania, Priyanshu Gupta, and Arjun Radhakrishna. Logic-lm++: Multi-step refinement for symbolic formulations. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ ACL 2024)*, pp. 56–63, 2024.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.