SATQuest: A Verifier for Logical Reasoning Evaluation and Reinforcement Fine-Tuning of LLMs

Anonymous ACL submission

Abstract

Large language models (LLMs) exhibit strong general reasoning, yet the community lacks *controllable, scalable, and verifiable* tools to analyze and improve these abilities. We present `SATQuest`, a verifier that generates diverse SAT-based reasoning tasks directly from Conjunctive Normal Form (CNF) instances and checks answers objectively with PySAT. `SATQuest` factorizes evaluation along three orthogonal dimensions—*instance*, *problem type*, and *question format*—enabling fine-grained, multi-dimensional analysis and reinforcement fine-tuning. Randomized CNF generation mitigates memorization and supports reproducible experiments. Using `SATQuest`, we benchmark a range of open- and closed-weight LLMs and uncover persistent gaps in logical reasoning, particularly on higher-complexity tasks and in transfer beyond familiar mathematical notation to machine or narrative formats. We further show that reinforcement fine-tuning with `SATQuest` rewards substantially boosts targeted performance and generalizes to larger instances, while cross-format robustness remains challenging. Collectively, `SATQuest` provides verifier-backed infrastructure for controlled, scalable, and reproducible *empirical research* on LLM logical reasoning and its training.

Anonymous GitHub

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable proficiency in general reasoning tasks, including complex problem-solving and code generation, with models like `o3-mini`(OpenAI, 2025), `DeepSeek-R1`(DeepSeek-AI et al., 2025), and `QwQ-32B`(Team, 2024) excelling in programming, mathematics, and scientific question-answering(Brown et al., 2020; OpenAI et al., 2024; Wei et al., 2022). This advanced reasoning capability, a cornerstone for Artificial General Intelligence (AGI), is a critical indicator of models' deep understanding and generalization.

To further explore the reasoning capabilities of LLMs, we urgently need evaluation and training tools that are both controllable and scalable. Fine-grained and reliable performance analysis requires systematic variable control, which is fundamental to empirical scientific research. However, existing benchmarks and datasets have significant limitations in variable control, making it difficult to support multi-dimensional, systematic analysis and training experiments—hindering deeper understanding of reasoning mechanisms in LLMs.

Although benchmarks such as *ZebraLogic* (Lin et al., 2025) and *Knights and Knaves* (Xie et al., 2024) have introduced controllable difficulty dimensions, their question types and formats remain narrow, supporting only the *instance* dimension. *MATH* (Hendrycks et al., 2021b) and *LiveBench* (White et al., 2025) provide rich content across mathematics and programming; however, they lack structured relationships and rely heavily on human judgment for difficulty labeling. In comparison, general-purpose benchmarks like *GPQA* (Rein et al., 2024), *MMLU* (Hendrycks et al., 2021a), and *Big-Bench* (Srivastava et al., 2023) offer broad coverage, yet suffer from issues such as data leakage, lack of continuity, and insufficient support for multi-dimensional controlled analysis.

In this work, we aim to construct a multi-dimensionally controllable and scalable verifier to support the evaluation and training of LLM reasoning capabilities, enabling the tracking of progress and discovery of limitations in LLM reasoning. Our main contributions are as follows:

(1) **CNF-native, controllable verifier.** We present `SATQuest`, which generates SAT-based reasoning tasks directly from CNF and organizes them along three orthogonal axes—*instance*, *problem type*, and *question format*. Randomized CNF construction, binary-string targets, and objective

PySAT checking enable leakage-resistant, verifiable evaluation and training; we also release 🤗 ANON/SATQuest for reproducibility.

(2) **Multi-dimensional evaluation.** Using SATQuest, we provide a comprehensive study of open/closed-weight LLMs across all three axes, revealing consistent failure modes—strong cross-format sensitivity beyond Math and accuracy degradation with solver-measured complexity—while surfacing hallucination under high complexity.

(3) **Verifier-driven Reinforcement Fine-tuning.** We implement reinforcement fine-tuning with SATQuest rewards, yielding targeted gains and longer reasoning chains. Improvements transfer to larger instances and exhibit *asymmetric* cross-task generalization, while cross-format robustness remains limited.

2 SATQuest Challenge Design

Overview. SATQuest is a systematic verifier engineered to comprehensively evaluate and enhance the logical reasoning capabilities of LLMs. Its primary goal is to offer a framework for fine-grained analysis, providing deeper insights into the strengths and limitations of LLMs in logical deduction. SATQuest is not designed to train LLMs as general-purpose solvers or to make them surpass specialized symbolic solvers in speed or accuracy.

To achieve its objectives, SATQuest automatically generates a variety of logical reasoning tasks directly derived from CNF instances. These tasks are meticulously organized along three orthogonal dimensions: *instance*, *problem type*, and *question format*, each targeting distinct aspects of logical reasoning. This multi-dimensional structure creates a comprehensive and controllable challenge space suitable for nuanced evaluation and effective reinforcement fine-tuning. CNF instances are utilized as the foundational elements due to their formal clarity, their established role as a standard representation in propositional logic, and their inherent compatibility with established SAT solvers for objective answer verification, making them an ideal medium for systematically probing the logical capabilities of LLMs.

Data Generation. SATQuest supports evaluation using any CNF instance dataset stored in the standard DIMACS format. For reproducibility, we generated two CNF datasets using the procedure outlined in Algorithm 1:

- 🤗 ANON/SATQuest: This dataset is generated for evaluation purposes (Sec. 3), specifically to assess logical reasoning across varying instance scales and difficulties. It consists of randomly generated CNF instances with $n \in [3, 16]$ variables and a fixed clause-to-variable ratio resulting in $m = 4n$ clauses. For each (n, m) configuration, 10 CNF instance pairs (one satisfiable and one unsatisfiable) were generated, resulting in a total of 140 CNF pairs. The two CNF instances share the same number of literals and nearly identical CNF structures.

- 🤗 ANON/SATQuest-RFT-3k: This dataset is generated for reinforcement fine-tuning (Sec. 4). It consists of CNF instances with $n \in [3, 8]$ variables and clause counts m determined by varying the clause-to-variable ratio from 2.1 to 4.0 in increments of 0.1 (i.e., m ranges from $2.1n$ to $4.0n$). For each (n, m) configuration, 25 CNF instance pairs were generated, resulting in a total of $3,000$ CNF pairs.

Challenge Dimensions 1: *Instance*. We categorize instances by their *scale* (number of variables n, clauses m, and literals) and inherent *difficulty*. Computational *difficulty* is assessed using SAT solver statistics: *decisions*, indicating search breadth; *conflicts*, reflecting constraint-driven backtracking; and *propagations*, quantifying chained logical implications. These structural and solver-derived metrics provide a multi-faceted characterization of an instance's combinatorial complexity, where higher values generally correspond to more challenging problems.

Challenge Dimensions 2: *Problem Type*. Let $\alpha : X \to \{0, 1\}$ be an assignment for $X = \{x_1, \ldots, x_n\}$, and let $F = \bigwedge_{i=1}^{m} C_i$ be a CNF formula. Each C_i is a clause of the form $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \cdots \vee \ell_{i,k_i}$, where $\ell_{i,j} \in X \cup \{\neg x | x \in X\}$. We define five fundamental SAT-based problems:

- SATDP (SAT Decision Problem): Determine whether F is satisfiable:

$$\mathsf{SATDP}(F) = \begin{cases} 1, & \exists\, \alpha \text{ s.t. } F(\alpha) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

  Tests the fundamental ability to determine logical consistency.

- SATSP (SAT Search Problem): If F is satisfiable, find an assignment α:

$$\alpha \text{ s.t. } F(\alpha) = 1.$$

Probes constructive reasoning by requiring the generation of a satisfying assignment.

- `MaxSAT` (Maximum Satisfiability): Find the assignment α^* that maximizes the number of satisfied clauses:

$$\alpha^* = \arg\max_{\alpha} \sum_{i=1}^{m} \mathbf{1}[C_i(\alpha) = 1].$$

Evaluates optimization skills when maximizing clause satisfaction under conflicting constraints.

- `MCS` (Minimal Correction Subset): For an unsatisfiable F, find a minimal set $S \subseteq \{1, \ldots, m\}$ whose removal yields a satisfiable formula:

$$\texttt{SATDP}(\bigwedge_{i \in S} C_i) = 1 \bigwedge \forall\, S' \subset S,\ \texttt{SATDP}(\bigwedge_{i \in S'} C_i) = 0.$$

Tests diagnostic reasoning through the identification of minimal corrections for unsatisfiability.

- `MUS` (Minimal Unsatisfiable Subset): For an unsatisfiable F, find a minimal unsatisfiable core $S \subseteq \{1, \ldots, m\}$:

$$\texttt{SATDP}(\bigwedge_{i \in S} C_i) = 0 \bigwedge \forall\, S' \subset S,\ \texttt{SATDP}(\bigwedge_{i \in S'} C_i) = 1.$$

Probes diagnostic reasoning by localizing minimal sources of logical inconsistency.

Each CNF instance pair consists of one satisfiable and one unsatisfiable formula. The satisfiable instance is used for `SATSP`, while the unsatisfiable instance is used for `MaxSAT`, `MCS`, and `MUS`. For `SATDP`, both instances are evaluated, forming two sub-tasks (`SATDP-sat` and `SATDP-unsat`). A response to `SATDP` is considered correct only if both sub-tasks are answered correctly, thereby discouraging random guessing. These problems progressively challenge LLM reasoning capabilities, from foundational deduction and solution construction (`SATDP`/`SATSP`), to constrained optimization (`MaxSAT`), and finally to minimal cause identification and correction (`MCS`/`MUS`).

Challenge Dimensions 3: *Question Format*. Recognizing that the presentation of a problem can significantly impact an LLM's reasoning process, this dimension introduces four logically equivalent representational formats for each CNF instance. This variation aims to test different reasoning skills and reduce reliance on superficial pattern matching.

- `Math` (mathematical notation): Uses $\wedge$, $\vee$, and $\neg$ to represent logic formulas. Balances between formality and readability.
  Example: $x_1 \vee \neg x_2 \vee x_3$

- `DIMACS` (machine format): A minimal, line-based format for representing Boolean formulas in CNF, the standard input for many SAT symbolic solvers.
  Example:

```
p cnf 3 1
1 -2 3 0
```

- `Story` (OR semantics, cookie day scenario): Wraps clauses as friendly narratives—"Alice is happy if..."—to test LLMs' ability to ground disjunctions in natural language.
  Example: "Alice will be happy if she gets crunchy choco (x_1), chewy vanilla ($\neg x_2$), or crunchy peanut (x_3)."

- `DualStory` (AND semantics, cookie day scenario): Presents the negated form—"Alice will be unhappy only if..." —turning OR into AND and requiring semantic tracking.
  Example: "Alice will be unhappy only if she is served crunchy choco (x_1), chewy vanilla ($\neg x_2$), and crunchy peanut (x_3)."

`Math` is common in training data and accessible to math-tuned LLMs, whereas `DIMACS` is a compact, noise-free, machine-readable format that tests a model's ability to interpret raw clause structures. It is specifically designed for evaluating LLMs without relying on mathematical training. `Story` and `DualStory` introduce narrative elements that add informational noise and require the model to translate natural-language logical structures into formal logic before reasoning.

Answer Verification. Prior logical reasoning benchmarks (e.g., (Lin et al., 2025; Xie et al., 2024; Mondorf and Plank, 2024)) typically require LLMs to produce `JSON` outputs, choose from preset options, or use complex extraction—methods that hinder automation and scalability despite reduced manual overhead. Moreover, multiple-choice formats

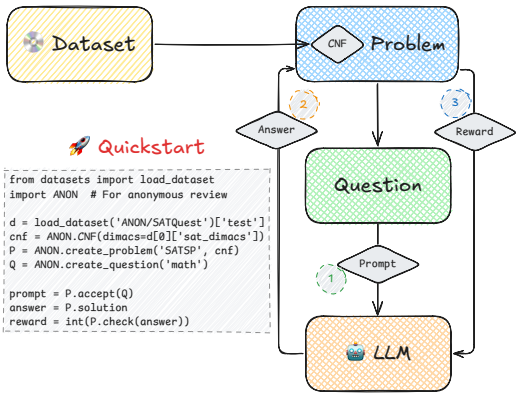

Figure 1: SATQuest Pipeline and Quickstart.

can significantly alter the true difficulty of problems, as the complexity of SATSP differs fundamentally from SAT verification. SATQuest simplifies evaluation by instructing LLMs to output binary strings—1 bit for SATDP, n bits for SATSP/MaxSAT, and m bits for MCS/MUS. These strings are extracted using regex and checked against CNF constraints via PySAT, allowing for multiple valid answers. A known limitation is that long binary outputs may challenge smaller models' ability to adhere to the expected format. To shed light on this, Fig. 7 presents format correctness statistics for mainstream models evaluated. For the complete prompt and detailed output format instructions, refer to App. B. The SATQuest Pipeline and Quickstart are illustrated in Fig. 1.

3 Evaluation

Overview. We conduct a comprehensive evaluation of the logical reasoning performance of various LLMs using the SATQuest benchmark. The analysis spans different *instance*, *problem types*, and *question formats*, enabling fine-grained, multi-dimensional insights. We begin with the overall benchmark results, followed by detailed analyses along each dimension.

Setup. We evaluate a diverse set of state-of-the-art open-weight and closed-weight LLMs, including vanilla models, reasoning models, and distilled variants. The evaluation uses the 🤗 ANON/SATQuest dataset, comprising 140 CNF instance pairs categorized by scale ($n \in [3, 16]$, clauses $m = 4n$). Each CNF pair yields tasks across five logical reasoning types (SATDP, SATSP, MaxSAT, MCS, MUS) and four question formats (Math, DIMACS, Story, DualStory), resulting in 20 evaluations per CNF pair. For detailed evaluation configurations and parameters, see App. E.

Overall Results. Fig. 2 shows model accuracies on SATQuest. o3-mini leads with 0.56 accuracy, followed by DeepSeek-R1 (0.42), QwQ-32B (0.40), and DeepSeek-R1-Distill-Qwen-32B (0.39), indicating that reasoning-enhanced models outperform vanilla LLMs. Large vanilla models like GPT-4.1 (0.38) and DeepSeek-V3-0324 (0.36) perform competitively despite lacking explicit reasoning training. In contrast, smaller vanilla models (e.g., Qwen2.5-7B-Instruct) achieve below 0.1 accuracy, revealing limited reasoning capabilities.

These results point to two notable trends. First, reasoning models consistently outperform vanilla counterparts, particularly on more complex tasks. Second, the modest overall accuracy across models reflects the challenging nature of SATQuest and its effectiveness in distinguishing reasoning capabilities. Moreover, we observe that performance on SATQuest is highly correlated with other recent reasoning benchmarks, such as *GPQA* (Rein et al., 2024) and *ZebraLogic* (Lin et al., 2025), suggesting that SATQuest captures essential generalization and reasoning capabilities in LLMs.

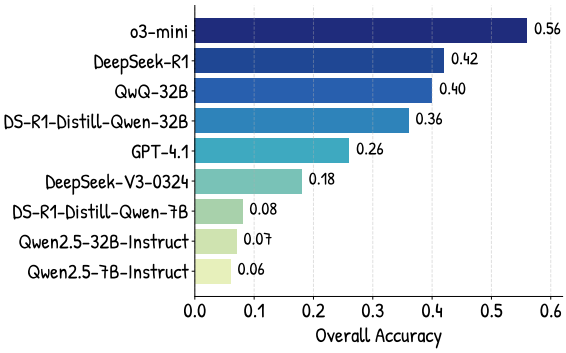

Figure 2: Overall accuracy of evaluated LLMs on the SATQuest benchmark, averaged across all *problem types* and *question formats*.

Analysis by *Instance*. We examine how model performance scales with instance complexity, measured using the number of decisions made by established SAT solvers (Glucose 4.1 (Audemard and Simon, 2018) for SATDP/SATSP, RC2 (Ignatiev et al., 2019) for MaxSAT, LBX (Mencía et al., 2015) for MCS, and MUSX (Marques-Silva, 2010) for MUS). Fig. 3 visualizes model accuracy and response length against this complexity metric. A consistent trend across tasks is that as instance complexity increases (more solver decisions), model accuracy tends to decline, while response length generally increases. Notably, we observe a concerning hallucination phenomenon when models encounter highly

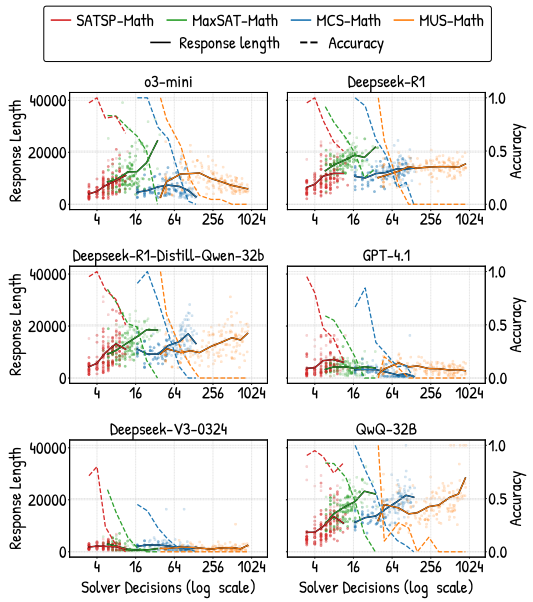

Figure 3: Accuracy (solid lines, right axis) and response length (dashed lines, left axis) vs. instance complexity (solver decisions) for SATSP-Math, MaxSAT-Math, MCS-Math and MUS-Math tasks.

complex instances: they often fabricate solver calls or invent simplified reasoning paths rather than engaging with the full logical complexity of the problem. This hallucination is particularly evident for o3-mini on MCS-Math and MUS-Math tasks, where response length actually decreases at high complexity, indicating the model abandons complete reasoning in favor of hallucinated shortcuts. Overall, all models exhibit reduced accuracy on larger and more difficult instances. Top-performing models like o3-mini and DeepSeek-R1 show more gradual degradation, indicating better scalability, whereas less capable models experience a sharp performance drop, often failing completely on moderately complex instances and resorting to increasingly severe hallucinations when faced with logical complexity beyond their reasoning capacity.

Analysis by *Problem Type*. We analyze performance variations across the five distinct SAT-based problem types: SATDP, SATSP, MaxSAT, MCS, and MUS. The heatmaps in Fig. 4 reveal performance differences across these tasks. Consistently across models, performance tends to be highest on SATDP (especially SATDP-sat) and SATSP, and lowest on MCS and MUS, forming a clear difficulty hierarchy. Models generally handle the basic decision (SATDP) and search (SATSP) tasks better than tasks requiring optimization or diagnosis, though performance on SATSP is often lower than SATDP. Performance on the optimization task MaxSAT typically sits between the basic tasks and the diagnostic tasks. The

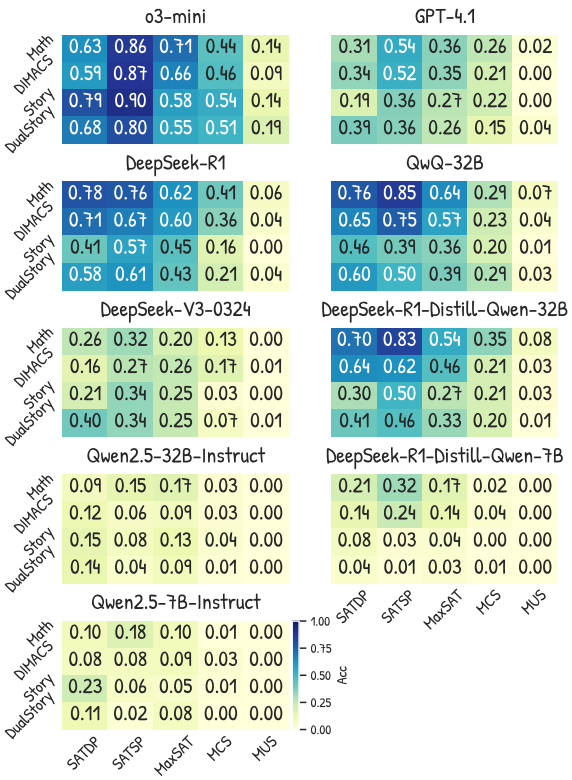

Figure 4: Accuracy heatmaps from a single run for each model, showing performance by *problem type* (columns) and *question format* (rows).

diagnostic tasks, MCS and MUS, which require identifying minimal subsets, prove particularly challenging, with accuracy dropping significantly for almost all models. While top models like o3-mini maintain some capability even on harder tasks, the gap between task types is pronounced across the board. This performance stratification aligns with the solver complexity shown in Fig. 3 (measured by solver decisions), suggesting LLMs struggle progressively more with tasks demanding global optimization, minimality constraints, and diagnostic reasoning over combinatorial spaces. Overall, the results highlight LLM limitations in tackling the full spectrum of logical reasoning challenges represented by these diverse SAT-based tasks.

Analysis by *Question Format*. The way a logical problem is presented can significantly affect an LLM's ability to solve it. We analyze this impact by evaluating performance across four distinct question formats: Math, DIMACS, Story, and DualStory. Fig. 4 illustrates how accuracy varies across these formats. All models perform best in the Math format, generally achieving their highest accuracy, followed by DIMACS, with Story and DualStory formats yielding the lowest accuracy. o3-mini demonstrates relatively stable performance across the four formats, indicating

strong reasoning robustness regardless of presentation style. However, other open-weight reasoning models like DeepSeek-R1 and QwQ-32B, while performing well in the Math format, exhibit a significant drop in accuracy in other formats, suggesting higher sensitivity to the presentation style.

It is noteworthy that the Story and DualStory problems introduce narrative elements, adding informational noise and requiring the model to translate the natural language logical structure into formal logic before reasoning. The increased difficulty and subsequent lower accuracy are thus expected. However, the DIMACS format is structurally similar to Math, contains no redundant information, and has higher information density. Despite this, open-weight reasoning models still show a marked decrease in accuracy compared to Math (e.g., DeepSeek-R1 and QwQ-32B accuracy dropped by 9% and 10% respectively on SATSP-DIMACS compared to SATSP-Math).

Through case studies presented in App. D, we observe that DeepSeek-R1 and QwQ-32B often attempt to reason directly within the DIMACS format rather than translating it into formal mathematical notation. This approach involves working with the raw DIMACS clauses, which requires tracking multiple variable assignments simultaneously across numerous constraints. The models frequently make errors when attempting to verify clause satisfaction or when determining the implications of specific variable assignments, particularly misinterpreting the disjunctive nature of clauses or conflating the semantic meaning of positive and negative literals. This direct approach appears to lead to a higher error rate during the reasoning process, as the models struggle to maintain consistency across the complex network of logical constraints represented in the DIMACS format.

Conversely, vanilla models like GPT-4.1 and DeepSeek-V3-0324, although performing worse overall, show relatively balanced performance across different formats, indicating lower format sensitivity. These models sometimes employ structured thinking approaches by first translating DIMACS inputs into the Math format before proceeding with reasoning, or by introducing meaningful symbolic notation during their reasoning process, which appears to enhance reasoning stability. The shorter reasoning chains produced by vanilla models may also contribute to their format robustness, as briefer deductions have fewer opportunities for errors to accumulate. This contrasts with DeepSeek-R1 and QwQ-32B, whose struggles outside the Math format seem to stem from relying more on potentially error-prone direct reasoning or trial-and-error within unfamiliar formats, rather than employing systematic format translation or structured analysis.

Achieving AGI likely requires LLMs to reason effectively across diverse formats, enabling the integration of knowledge from different domains and fostering more powerful, generalized reasoning capabilities. SATQuest thus serves as a valuable benchmark for assessing LLMs' adaptability and robustness in logical reasoning across various presentation styles.

4 Reinforcement Fine-Tuning

Overview. As demonstrated by the evaluation in Sec. 3, current LLMs exhibit significant limitations in logical reasoning, with notable deficiencies in generalization across *instance*, *problem type*, and *question format*. This section explores avenues for enhancing LLM logical reasoning capabilities through Reinforcement Fine-Tuning (RFT), directly utilizing reward signals from the SATQuest verifier. Our investigation particularly focuses on two aspects: First, we assess whether SATQuest-driven RFT can stimulate LLMs to construct longer reasoning chains, thereby fostering deeper logical deduction. Second, we delve into the relationship between RFT and the generalization deficiencies identified in Sec. 3, aiming to elucidate the specific effects and potential bottlenecks of RFT in improving cross-task and cross-format generalization.

Setup. We select the Qwen2.5-7B-Instruct model as the baseline, utilizing the Group Relative Policy Optimization (GRPO) algorithm (Shao and et al., 2024) during fine-tuning. This baseline model is nearly a blank slate, a small vanilla model that only shows marginal performance on SATSP-Math. The training leverages the 🤗 ANON/SATQuest-RFT-3k dataset, comprising $3,000$ CNF instance pairs with $n \in [3, 8]$ variables and clause-to-variable ratios ranging from 2.1 to 4.0. We train three distinct models, each focusing on a specific *problem type* and *question format* combination: SATSP-Math, SATSP-Story, MaxSAT-Math.

Our implementation is based on the TRL library (von Werra et al., 2020), adopting the GRPO objective as described in (Shao and et al., 2024). The prompt template follows the structure pro-

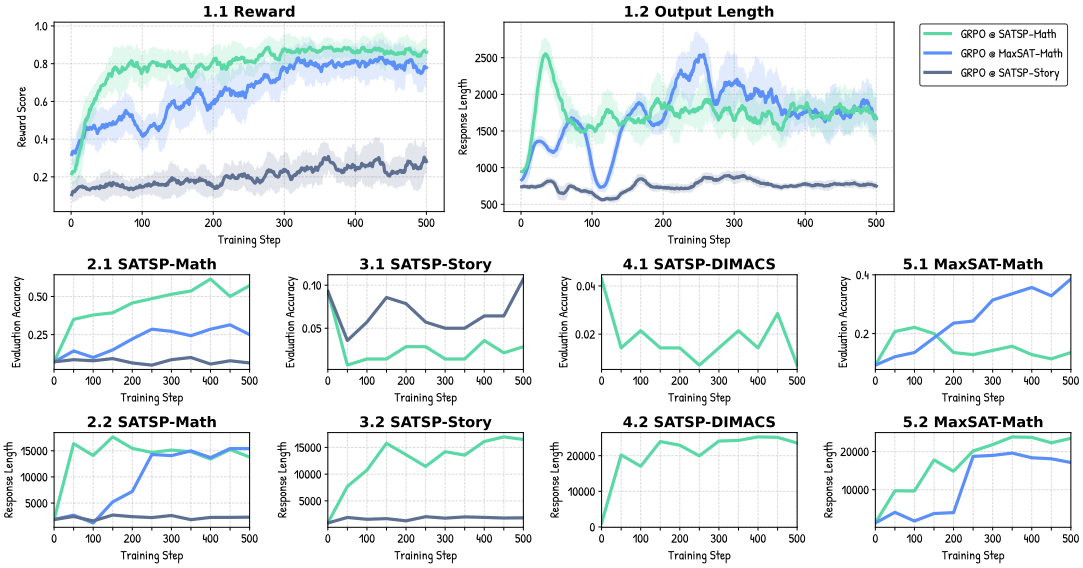

Figure 5: GRPO fine-tuning using `SATQuest` rewards. Training dynamics (Top: reward, response length) and evaluation performance (Bottom: accuracy, response length on target and generalization tasks) vs. training steps for models fine-tuned on `SATSP-Math`, `MaxSAT-Math`, and `SATSP-Story`. Solid lines show the mean; shaded regions denote ± 1 standard deviation.

posed in (DeepSeek-AI et al., 2025). The reward function is designed using the `SATQuest` verifier, assigning a reward of 1.0 for correct answers and 0.0 for incorrect ones. Additionally, two format correctness rewards, detailed in App. F.1, are incorporated with weights of 0.05 each, complementing the primary reward weight of 1.0. For training we set max_prompt_length to 2048 and $max_completion_length$ to 8192, for evaluation we set $max_prompt_length + max_completion_length$ to 32768. The training and evaluation parameters are detailed in App. F.2.

Training Dynamics. Training curves (Fig. 5, top row) reveal that models trained on `Math`-based tasks (`SATSP-Math` and `MaxSAT-Math`) achieve higher rewards and generate longer responses than those trained on `SATSP-Story`. The `Math` format appears to better facilitate extended reasoning chains that receive positive reinforcement from the verifier. Our `SATQuest` verifier effectively stimulates extended reasoning development within few training steps, especially with the mathematical format. Response length curves show distinct patterns—rapid initial growth as models learn longer reasoning chains, temporary decline when adapting to format constraints at the training response limit (8192 tokens), followed by stabilization.

Training on `SATSP-Story` proved less effective, largely due to the baseline model's weak narrative reasoning abilities. While *Logic-RL* (Xie et al., 2025) has successfully stimulated narrative reasoning, our tasks involve substantially higher complexity and scale. Convergence efficiency correlates with task complexity and initial model capabilities, though these factors require further investigation to fully separate.

Generalization Across *Instance*. We observe positive generalization concerning problem complexity. Models fine-tuned on `SATSP-Math` and `MaxSAT-Math` demonstrated improved accuracy when evaluated on the corresponding tasks within the evaluation set (Fig. 5, subplots 2.1, 5.1). Crucially, these evaluation instances involved larger scales ($n > 8$) than those used during training ($n \in [3, 8]$), indicating that the learned reasoning skills generalize to more complex instances within the same problem and format.

Generalization Across *Problem Types*. Our results reveal an interesting asymmetry in cross-problem generalization. Fine-tuning on the more complex `MaxSAT-Math` task led to performance improvements not only on `MaxSAT-Math` itself but also conferred benefits to the simpler `SATSP-Math` task. However, the model trained solely on `SATSP-Math` did not show a corresponding improvement on `MaxSAT-Math` (compare improvements patterns in Fig. 5, bottom row). This suggests that the reasoning capabilities required for `MaxSAT` may encompass those needed for `SATSP`. Strategically, this implies that training on more complex and diverse logical problems could be more effective for fostering robust reasoning skills that generalize to simpler, related problems.

Generalization Type	Positive Transfer?	Observation
Across Instances	✓	Improved accuracy on larger unseen CNF instances.
Across Problem Types	✓ (Asymmetric)	MaxSAT → SATSP transfer stronger.
Across Question Formats	✗	Math → DIMACS/Story weak transfer.

Table 1: Summary of generalization findings across SATQuest dimensions.

Generalization Across *Question Formats*. Cross-format generalization remains notably difficult. The model fine-tuned on SATSP-Math shows minimal improvement when evaluated on other formats such as SATSP-Story (Fig. 5, subplot 3.1) and even on the structurally similar SATSP-DIMACS task (subplot 4.1). This suggests that reasoning capabilities acquired in the Math format do not readily transfer to logically equivalent tasks presented in alternative formats, whether narrative or machine-readable. Further analysis of failure cases reveals that, after SATSP-Math fine-tuning, the model tends to generate verbose but flawed reasoning when confronted with DIMACS inputs—mirroring the issues described in Sec. 3 and App. D. The model appears to have overfitted to a specific Math-style reasoning pattern, at the expense of its initial structured thinking ability, and fails to effectively translate DIMACS representations into a suitable reasoning form. These findings indicate that small-scale RFT may not be sufficient to overcome format generalization barriers, and that the performance discrepancies across formats observed in Sec. 3 may partially stem from the limitations of format-specific fine-tuning itself.

5 Related Work

Researchers have developed numerous benchmarks to evaluate LLMs' capabilities. Popular evaluations like (Rein et al., 2024; Hendrycks et al., 2021b,a; Srivastava et al., 2023) comprehensively assess LLMs but suffer from data leakage and lack of continuity, often being solved by advanced models within 18 months of introduction. While newer benchmarks (Suzgun et al., 2023; Kazemi et al., 2025; Team et al., 2025; Gema et al., 2025; Wang et al., 2024; Glazer et al., 2024) offer improvements, core issues remain. White et al. (2025) introduced dynamic question banks and automatic scoring but still relies on manual difficulty annotation without multi-dimensional analysis controls. These evaluations primarily assess capabilities rather than providing insights into internal mechanisms. Some studies explore more nuanced approaches: Lin et al. (2025); Xie et al. (2024) use formalized templates with controllable difficulty dimensions for finer-grained analysis. Xie et al. (2025) conducts RL training on *K&K* to investigate how RL enhances reasoning capabilities. He et al. (2024) provides evaluation through multilingual coverage and multi-turn design. Huang et al. (2025); Yu et al. (2025) test mathematical reasoning robustness through minimal perturbations. Research by Hazra et al. (2025) investigates LLM reasoning capabilities through 3-SAT phase transitions. Our work, SATQuest, uses randomly generated CNF instances to prevent data leakage and ensure continuity. We provide five interrelated SAT-based *problem types* and four *question formats* with different information densities. These three orthogonal dimensions—*instance*, *problem type*, and *question format*—enable flexible experimental control for future LLM reasoning research.

6 Conclusion

We introduce SATQuest, a CNF-native verifier that systematically structures SAT-based reasoning evaluation and training along three orthogonal axes: *instance*, *problem type*, and *question format*. Our results reveal that while targeted reinforcement fine-tuning with verifiable rewards enables measurable gains—especially for harder or larger instances—persistent weaknesses remain in diagnostic reasoning and robustness to presentation changes. The key patterns of generalization and their asymmetries are summarized in Table 1.

Takeaway. SATQuest provides a controllable and verifiable feedback loop for probing LLM logical reasoning. It reliably exposes non-obvious failure modes—particularly format sensitivity and difficulty with minimal diagnoses—while enabling principled improvements without manual heuristics. Future progress will require richer reward signals and explicit curriculum bridging across formats and domains, to close the gap between instance-specific improvements and generalizable, format-robust reasoning, and we release datasets and code to catalyze progress.

Limitations

Scope and intent. SATQuest is designed as a *controllable, verifiable, and factorized* probe of LLM logical reasoning along three axes (*instance / problem type / question format*), prioritizing clean manipulation of variables over breadth. MCS/MUS are general diagnostic notions beyond propositional logic, but in this work they are *instantiated* as clause-level subsets over CNF; richer logics (e.g., first-order, modal), multi-tool interaction, and open-ended natural-language tasks are out of scope. The evaluation set fixes $m{=}4n$ with $n{\leq}16$, the RFT training set uses $n{\leq}8$, and narratives are synthetic and English-only. Hence our conclusions target *relative* patterns across instance/problem/format rather than absolute real-world capability.

Technical. By design, our setup prioritizes *verifiable, controlled analysis*: models must return binary strings that we extract and check against the CNF with PySAT, which removes grading ambiguity but constrains the response space and can be brittle for long clause-mask outputs. All evaluations are conducted in a *strict zero-shot* regime using standardized prompts (no tools, no few-shot exemplars), to isolate reasoning from prompt engineering; the prompt templates used for evaluation are provided in the paper and appendix. On the training side, we only instantiate reinforcement fine-tuning with the GRPO algorithm, optimizing a *binary* (0/1) verifier reward; alternative policy-optimization methods, decoding strategies, and shaped rewards are not explored here. Task difficulty is proxied by classical solver statistics—decisions, conflicts, and propagations—which we compute with established solvers per task; the complete per-instance statistics and the solver identifiers are provided with the dataset metadata.

Future work. Future directions include expanding the diversity and realism of question formats—incorporating both LLM-generated and programmatically varied representations such as tabular, code-based, or diagrammatic/logical schemas—to better capture the heterogeneity of human reasoning contexts. We also plan to curate *industrial-scale CNF corpora* drawn from real verification, planning, and scheduling benchmarks with standardized metadata for open research use. On the learning side, designing *dense and shaped rewards* (e.g., clause coverage, unit-propagation progress, MCS/MUS cardinality margins, or partial-assignment consistency) may provide more informative optimization signals than binary correctness alone. Curriculum learning across scale (n, m), problem-type mixtures, and structural regimes can help bridge small synthetic tasks and large real-world instances. Finally, SATQuest is intended to serve as a practical foundation for *empirical research* on LLM logical reasoning—offering a controllable, verifier-backed testbed for standardized and reproducible studies of reward shaping, curricula, and cross-format transfer.

References

Gilles Audemard and Laurent Simon. 2018. On the glucose sat solver. *International Journal on Artificial Intelligence Tools*, 27(01):1840001.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile Van Krieken, and Pasquale Minervini. 2025. Are we done with MMLU? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5069–5096, Albuquerque, New Mexico. Association for Computational Linguistics.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, and 5 others. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *Preprint*, arXiv:2411.04872.

Rishi Hazra, Gabriele Venturato, Pedro Zuidberg Dos Martires, and Luc De Raedt. 2025. Have large

language models learned to reason? a characterization via 3-sat phase transition. *Preprint*, arXiv:2504.03930.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, Shruti Bhosale, Chenguang Zhu, Karthik Abinav Sankararaman, Eryk Helenowski, Melanie Kambadur, Aditya Tayade, Hao Ma, Han Fang, and Sinong Wang. 2024. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following. *Preprint*, arXiv:2410.15553.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Shulin Huang, Linyi Yang, Yan Song, Shuang Chen, Leyang Cui, Ziyu Wan, Qingcheng Zeng, Ying Wen, Kun Shao, Weinan Zhang, Jun Wang, and Yue Zhang. 2025. Thinkbench: Dynamic out-of-distribution evaluation for robust llm reasoning. *Preprint*, arXiv:2502.16268.

Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. 2019. Rc2: an efficient maxsat solver. *Journal on Satisfiability, Boolean Modelling and Computation*, 11(1):53–64.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V. Le, and Orhan Firat. 2025. Big-bench extra hard. *Preprint*, arXiv:2502.19187.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. 2025. Zebralogic: On the scaling limits of llms for logical reasoning. *Preprint*, arXiv:2502.01100.

Joao Marques-Silva. 2010. Minimal unsatisfiability: Models, algorithms and applications (invited paper). In *Proceedings of the 2010 40th IEEE International Symposium on Multiple-Valued Logic*, ISMVL '10, page 9–14, USA. IEEE Computer Society.

Carlos Mencía, Alessandro Previti, and Joao Marques-Silva. 2015. Literal-based mcs extraction. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1973–1979. AAAI Press.

Philipp Mondorf and Barbara Plank. 2024. Liar, liar, logical mire: A benchmark for suppositional reasoning in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7114–7137, Miami, Florida, USA. Association for Computational Linguistics.

OpenAI. 2025. OpenAI o3-mini System Card. `https://cdn.openai.com/o3-mini-system-card-feb10.pdf`. Accessed: 2025-04-22.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. 2019. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations*.

Zhihong Shao and et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 430 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. Featured Certification.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

P Team, Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixin Deng, Shawn Gavin, Shian Jia, Sichao Jiang, Yiyan Liao, Rui Li, Qinrui Li, and 78 others. 2025. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *Preprint*, arXiv:2502.14739.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2025. Livebench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations*.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2024. On memorization of large language models in logical reasoning. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *Preprint*, arXiv:2502.14768.

Tong Yu, Yongcheng Jing, Xikun Zhang, Wentao Jiang, Wenjie Wu, Yingjie Wang, Wenbin Hu, Bo Du, and Dacheng Tao. 2025. Benchmarking reasoning robustness in large language models. *Preprint*, arXiv:2503.04550.

A Generate CNF Pair

The CNF generation algorithm is adapted from (Selsam et al., 2019)[1].

Algorithm 1 Generate CNF Pair (Satisfiable and Unsatisfiable)

```
procedure GenCNFPair(n, m, p_{k_2}, p_{geo})
    unsat_clauses ← ∅
    while solve(unsat_clauses) do                          ▷ Generate unsatisfiable CNF
        unsat_clauses ← ∅
        while |unsat| < m do
            k ← 1 if rand() < p_{k_2} else 2 + Geometric(p_{geo})
            k ← min(k, n)
            clause ← RandomClause(n, k)                    ▷ Random literals with polarity
            unsat_clauses ← unsat_clauses ∪ {clause}
        end while
    end while
    sat_clauses ← unsat_clauses
    while ¬solve(sat_clauses) do                           ▷ Convert to satisfiable CNF
        sat_clauses ← RandomFlipClause(sat_clauses)        ▷ Flip literal polarities
    end while
    return unsat_clauses, sat_clauses
end procedure
```

[1] https://github.com/dselsam/neurosat/blob/master/python/gen_sr_dimacs.py

B Prompt Details

In this appendix, we present the complete prompt templates and representative example outputs used to instruct LLMs to produce binary string responses for SAT-based reasoning tasks. These templates specify the precise wording, formatting requirements, and task descriptions used during both evaluation and reinforcement fine-tuning phases. Our prompt construction strategy adapts and extends established templates from OpenAI's `simple-eval`[2] and `DeepSeek-R1` (DeepSeek-AI et al., 2025). The following subsections detail how we systematically assemble task prompts and incorporate system-level configurations for both evaluation (Sec. 3) and fine-tuning (Sec. 4) experiments. Fig. 6 illustrates the general structure of the task prompt used throughout our experiments.

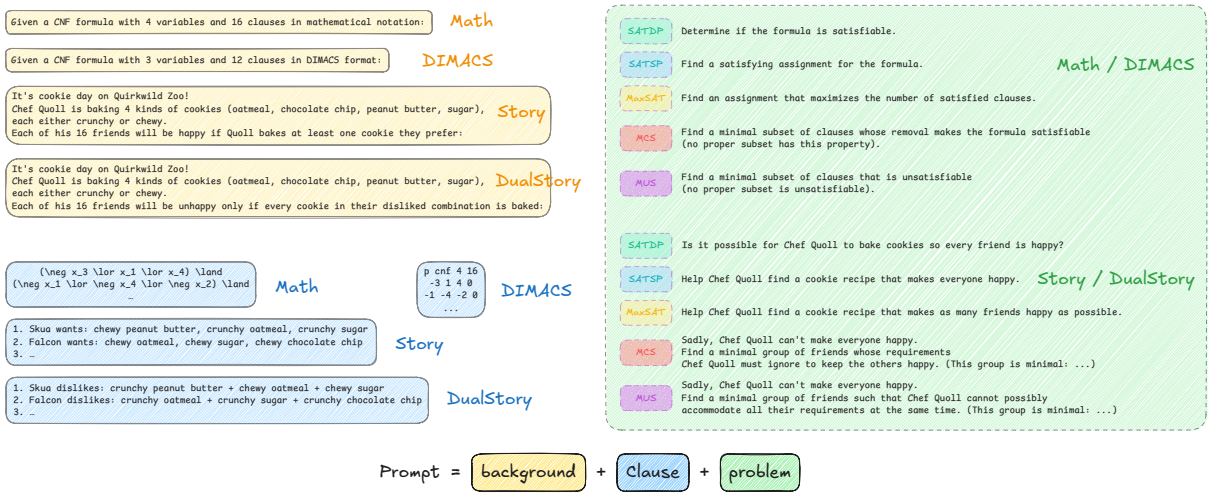

Figure 6: Task Prompt Construction in `SATQuest`

B.1 Evaluation Prompt Template

In Sec. 3, we employ the following prompt template that includes Chain-of-Thought (CoT) guidance and specific output format instructions to ensure consistent responses across all models.

For vanilla model, add system prompt: `"You are a helpful assistant."`.

```
Solve the following problem step by step. The last line of your response should be of the form Answer: $ANSWER (without
    ↪ quotes) where $ANSWER is the answer to the problem.

Given a CNF formula with 4 variables and 16 clauses in DIMACS format:

p cnf 4 16
-3 1 4 0
-1 -4 -2 0
-1 4 2 0
2 -1 -4 0
2 -3 4 0
-3 -4 1 0
-4 1 -3 0
1 2 -4 0
-3 -2 1 0
4 -3 1 0
-1 -3 2 0
2 -3 4 0
-1 -2 3 0
2 3 4 0
2 3 1 0
1 3 -4 0

Find a satisfying assignment for the formula.
Output a binary string of length 4 ('1' for true, '0' for false).

Remember to put your answer on its own line after "Answer:", and you do not need to use a \boxed command.
```

[2] https://github.com/openai/simple-evals/blob/0f2cf3/math_eval.py

B.2 Reinforcement Fine-Tuning Prompt Template

For Sec. 4, we adapt the prompt to encourage explicit CoT reasoning by combining system instructions with a structured user message format.

```
<|im_start|>system
You are a helpful AI Assistant that provides well-reasoned and detailed responses. You first think about the reasoning
      ↪ process as an internal monologue and then provide the user with the answer. Respond in the following format:
      ↪ <think>\n...\n</think>\n<answer>\n...\n</answer>
<|im_end|>
<|im_start|>user
Given a CNF formula with 4 variables and 16 clauses in DIMACS format:

p cnf 4 16
-3 1 4 0
-1 -4 -2 0
-1 4 2 0
2 -1 -4 0
2 -3 4 0
-3 -4 1 0
-4 1 -3 0
1 2 -4 0
-3 -2 1 0
4 -3 1 0
-1 -3 2 0
2 -3 4 0
-1 -2 3 0
2 3 4 0
2 3 1 0
1 3 -4 0

Find a satisfying assignment for the formula.
Output a binary string of length 4 ('1' for true, '0' for false).

Show your work in <think> </think> tags. And return the final answer in <answer> </answer> tags, for example <answer> 0101
      ↪ </answer>.
<|im_end|>
<|im_start|>assistant
```

C Prompt Examples

For more prompt examples, please refer to the https://sat-quest-anon.vercel.app/examples/#prompt-gallery website.

SATSP-Math Prompt Example

```
Given a CNF formula with 3 variables and 4 clauses in mathematical notation:

(x_1 \lor \neg x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2) \land (\neg x_3)

Find a satisfying assignment for the formula.
Output a binary string of length 3 ('1' for true, '0' for false).
```

SATSP-DIMACS Prompt Example

```
Given a CNF formula with 3 variables and 4 clauses in DIMACS format:

p cnf 3 4
1 -2 3 0
-1 2 3 0
1 2 0
-3 0

Find a satisfying assignment for the formula.
Output a binary string of length 3 ('1' for true, '0' for false).
```

SATSP-Story Prompt Example

```
It's cookie day on Quirkwild Zoo!
Chef Orion is baking 3 kinds of cookies (chocolate comet, maple moon, ginger nebula), each either crunchy or chewy.
Each of his 4 friends will be happy if Orion bakes at least one cookie they prefer:


1. Aquila wants: crunchy chocolate comet, chewy maple moon, crunchy ginger nebula
2. Borealis wants: chewy chocolate comet, crunchy maple moon, crunchy ginger nebula
3. Cygnus wants: crunchy chocolate comet, crunchy maple moon
4. Draco wants: chewy ginger nebula

Find a satisfying assignment for the formula.
Output a binary string of length 3 ('1' for true, '0' for false).
```

SATSP-DualStory Prompt Example

```
It's cookie day on Quirkwild Zoo!
Chef Orion is baking 3 kinds of cookies (chocolate comet, maple moon, ginger nebula), each either crunchy or chewy.
Each of his 4 friends will be unhappy only if every cookie in their disliked combination is baked:


1. Aquila dislikes: chewy chocolate comet + crunchy maple moon + chewy ginger nebula
2. Borealis dislikes: crunchy chocolate comet + chewy maple moon + chewy ginger nebula
3. Cygnus dislikes: chewy chocolate comet + chewy maple moon
4. Draco dislikes: crunchy ginger nebula

Find a satisfying assignment for the formula.
Output a binary string of length 3 ('1' for true, '0' for false).
```

D Case Studies

Our analysis reveals that both DeepSeek-R1 and QwQ-32B predominantly reason directly with the DIMACS format rather than translating it into formal mathematical notation. Throughout the four case studies, segments highlighted in green demonstrate instances where the models correctly comprehend the CNF structure, especially regarding the disjunctive (OR) semantics within clauses. Conversely, red-highlighted sections expose critical reasoning flaws: models frequently misinterpret the disjunctive nature of clauses or conflate the semantic meaning of positive and negative literals (representing true versus false values). These observations suggest that direct reasoning within the DIMACS representation, without proper symbolic abstraction, frequently results in semantic drift and format-specific reasoning errors.

Case1: DeepSeek-R1 @ SATSP-DIMACS

```
Okay, so I need to find a satisfying assignment for this CNF formula with 9 variables and 36 clauses.

...

So, for each clause, at least one of the literals must be true in the assignment.

...

For each variable, check if all occurrences are positive, all negative, or mixed.

Let's start with variable 1:

Looking through the clauses where variable 1 appears.

Clause 5: -1 2 -6 5 8. So -1 here.

Clause 6:8 1 9 -6. Positive 1.

...

Clause1: -2 -9 6 -> with 6=0, this becomes -2 AND -9 -> so both must be true. So variables2 and 9 must be 0.

...

All clauses seem to be satisfied. So the assignment is valid.
```

Full prompt and response: [Link hidden for anonymity]

Case2: DeepSeek-R1 @ SATSP-DIMACS

```
Okay, so I need to find a satisfying assignment for this CNF formula with 10 variables and 40 clauses. Hmm, let's think.
    ↪ The variables are numbered 1 through 10, and each line in the DIMACS format represents a clause.
The clauses are ORs of literals, and the entire formula is the AND of all these clauses.
So the assignment has to make at least one literal in each clause true.

...

Clause 5: -1 -> negative occurrence.

Clause 6: 1 -> positive.

...

Ok, moving on.

Clause28:-5 -1 -> both must be false. So variable5=0 and variable1=0.

Wait, ...

All other clauses seem to have been handled earlier. So this assignment should work. Thus, the binary string is 0001010100.
```

Full prompt and response: [Link hidden for anonymity]

Case3: QwQ-32B @ SATSP-DIMACS

```
Okay, so I have this CNF formula with 9 variables and 36 clauses, and I need to find a satisfying assignment.

...

Let me start by writing down some of the clauses:

1. 5 3 -4 9 0 -> variables 5,3 are true, 4 is false, 9 is true.

2. -1 -2 7 -6 0 -> variables 1 and 2 are false, 7 and 6 are false?

Wait, no, the clause is satisfied if at least one literal is true.
So for this clause, either -1 (1 is false), -2 (2 false), 7 (true), or -6 (6 false). So ...

...

So this should be a valid assignment.
```

Full prompt and response: [Link hidden for anonymity]

Case4: QwQ-32B @ SATSP-DIMACS

```
Okay, so I need to find a satisfying assignment for this CNF formula with 10 variables and 40 clauses. Hmm, that's a bit
    ↪ of a challenge.
Let me think about how to approach this.
Since it's a CNF formula, each clause has to be satisfied by at least one literal being true.

...

Looking at the clauses:

The first few clauses are:

-9 4 -10 0 -> clause 1

6 -2 -8 -1 0 -> clause 2

....

Clause4: -1 -2 -4 -> since variables1 and2 are 0, this clause requires -4 (so variable4 must be 0).

...

Yes, that works.

Thus, the satisfying assignment is 1000101101.
```

Full prompt and response: [Link hidden for anonymity]

E Evaluation Configs

In all experiments in Sec. 3, we use APIs for evaluation. The specific model APIs and IDs are listed in Table 2, and the parameters for vanilla models are shown in Table 3. For reasoning models, we use their default configurations.

Table 2: Model API / ID of our Evaluated Models.

Model Name	API / ID
o3-mini	Azure API[3]: o3-mini-2025-01-31
GPT-4.1	Azure API: gpt-4.1-2025-04-14
DeepSeek-R1	VolcEngine API[4]: deepseek-r1
DeepSeek-V3-0324	VolcEngine API: deepseek-v3-0324
DeepSeek-R1-Distill-Qwen-7B	VolcEngine API: deepseek-r1-distill-qwen-7b
DeepSeek-R1-Distill-Qwen-32B	VolcEngine API: deepseek-r1-distill-qwen-32b
QwQ-32B	Alibaba Cloud API: qwq-32b-plus[5]
Qwen2.5-7B-Instruct	Alibaba Cloud API: qwen2.5-7b-instruct
Qwen2.5-32B-Instruct	Alibaba Cloud API: qwen2.5-32b-instruct

Table 3: Evaluation Parameters for Vanilla Models

Parameter	Value
temperature	0.6
top_p	1.0
max_tokens	16384

[3] *Azure OpenAI API*: https://azure.microsoft.com/en-us/products/ai-services/openai-service
[4] *ByteDance VolcEngine AI platform API*: https://www.volcengine.com/
[5] *Alibaba Cloud API*: https://www.alibabacloud.com/en

F Reinforcement Fine-Tuning Configs

F.1 Format Reward Functions

The format reward functions are adapted from the `huggingface/Open-R1` library[6].

```
def tag_count_reward(completions, **kwargs) -> list[float]:
    def count_tags(text: str) -> float:
        count = 0.0
        if text.count("<think>") == 1:
            count += 0.25
        if text.count("</think>") == 1:
            count += 0.25
        if text.count("<answer>") == 1:
            count += 0.25
        if text.count("</answer>") == 1:
            count += 0.25
        return count

    contents = [completion[0]["content"] for completion in completions]
    return [count_tags(c) for c in contents]

def format_reward(completions, **kwargs):
    _PATTERN = re.compile(r"<think>.*?</think>\s?<answer>.*?</answer>", flags=re.DOTALL)
    completion_contents = [completion[0]["content"] for completion in completions]
    rewards = []
    for c in completion_contents:
        text = str(c)
        total_len = len(text)
        if total_len == 0:
            rewards.append(0.0)
            continue

        m = _PATTERN.search(text)
        match_len = len(m.group()) if m else 0
        rewards.append(match_len / total_len)

    return rewards
```

F.2 Training Parameters

The training parameters used for GRPO of the `Qwen2.5-7B-Instruct` model are summarized in Table 4.

Table 4: Training Parameters for GRPO

Parameter	Value
`learning_rate`	0.000002
`batch_size`	`num_generations` $\times 8 = 128$
`max_grad_norm`	0.3
`num_iterations`	1
`beta`	0.01
`max_steps`	500
`max_prompt_length`	2048
`max_completion_length`	8192
`mask_truncated_completions`	`True`
`num_generations`	16
`temperature`	1.0
`scale_rewards`	`True`

F.3 Experiments Compute Resources

All experiments were conducted on a single server node equipped with 8 `NVIDIA A100 80GB` GPUs, 2 `Intel Xeon Platinum 8350C` CPU, and `1600GB` memory. We allocated 4 GPUs for training and 4 GPUs for *VLLM* inference. The training time for `GRPO@SATSP-Math`, `GRPO@MaxSAT-Math`, and `GRPO@SATSP-Story` was approximately 30 hours, 26 hours, and 9 hours, respectively.

[6] https://github.com/huggingface/open-r1/blob/main/src/open_r1/rewards.py

G Additional Figures

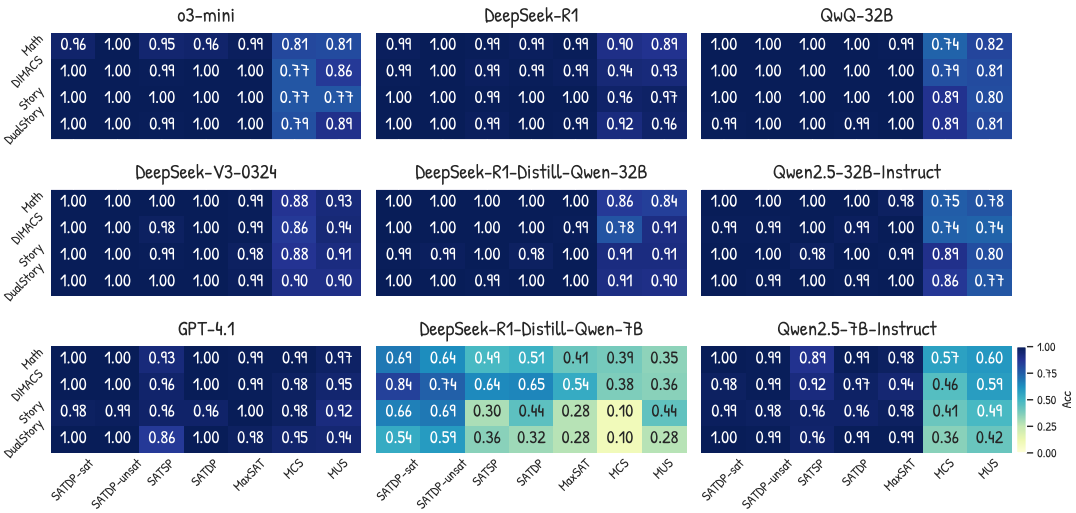

Figure 7: Format accuracy heatmaps showing LLM performance breakdown by *problem type* (columns) and *question format* (rows).

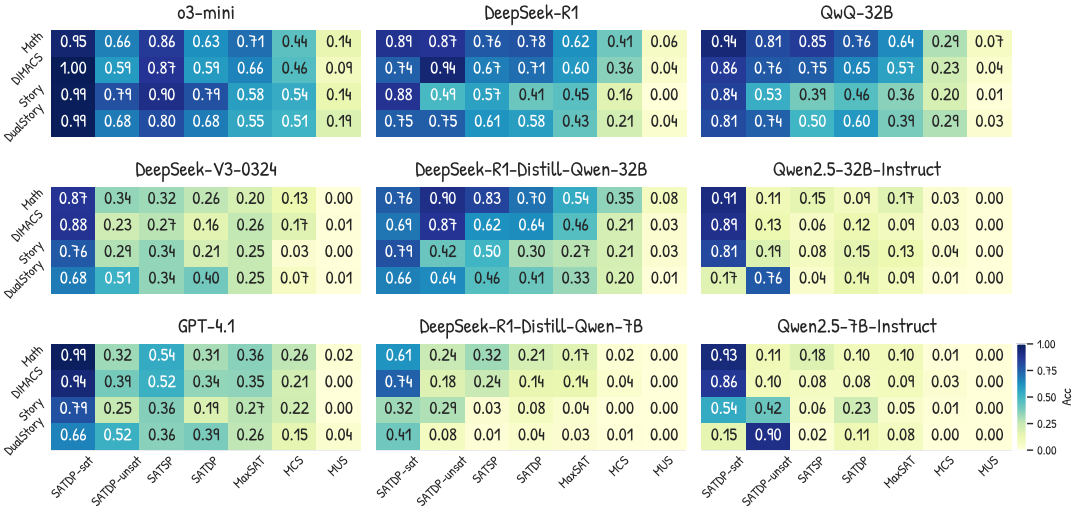

Figure 8: All accuracy heatmaps (including `SATDP-sat` and `SATDP-unsat`) showing LLM performance breakdown by *problem type* (columns) and *question format* (rows).

H Additional Experiments

Scaling to Larger Models. We applied the same RFT protocol to `Qwen2.5-14B-Instruct`, following the GRPO setup used for the 7B model. As shown in Table 5, the 14B model exhibits consistent improvements on all in-domain tasks, confirming `SATQuest`'s scalability. However, the cross-format generalization gap persists, suggesting that the limitation stems from current RFT methods rather than model size.

Model	Evaluation on `SATSP-Math`	Evaluation on `SATSP-Story`	Evaluation on `MaxSAT-Math`
`Qwen2.5-14B-Instruct`	12.86 ± 1.91	4.38 ± 1.77	12.94 ± 1.35
+ GRPO@`SATSP-Math`	70.80 ± 3.14	0.63 ± 0.80	17.05 ± 1.59
+ GRPO@`SATSP-Story`	11.96 ± 2.83	11.25 ± 1.31	14.20 ± 3.18
+ GRPO@`MaxSAT-Math`	41.52 ± 2.30	1.25 ± 0.63	46.33 ± 2.24
+ GRPO@`SATSP-Math/Story`	64.38 ± 3.71	10.54 ± 2.08	21.96 ± 2.88

Table 5: Additional RFT results on `Qwen2.5-14B-Instruct` (mean $\pm$ std over 8 runs).

Generalization Beyond SATQuest. To examine whether `SATQuest` training benefits broader reasoning tasks, we evaluated fine-tuned models on two external benchmarks: *AIME2024* (pass@4) and *DeepMind Mathematics*. As summarized in Table 6, `SATQuest`-trained models show consistent improvements across both datasets, indicating enhanced mathematical reasoning rather than overfitting to `SATQuest` formats.

Model	*AIME2024* (pass@4)	*DeepMind Mathematics*
`Qwen2.5-14B-Instruct`	24.17 ± 5.69	75.70 ± 0.74
+ GRPO@`SATSP-Math`	25.83 ± 3.19	79.15 ± 0.95
+ GRPO@`SATSP-Story`	20.84 ± 4.19	79.15 ± 0.55
+ GRPO@`MaxSAT-Math`	24.16 ± 1.67	77.72 ± 0.49
+ GRPO@`SATSP-Math/Story`	25.00 ± 4.30	79.30 ± 0.55

Table 6: Out-of-domain generalization on external reasoning benchmarks.

Summary. These extended results confirm that: (1) `SATQuest`-derived RFT scales to larger models; (2) improvements extend to independent reasoning benchmarks; and (3) the remaining cross-format gap reflects an intrinsic challenge in current RFT methods rather than dataset bias or model capacity.