LOST IN REAL-WORLD SCENARIOS: CONCRETIZATION DISRUPTS LLM LOGICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Although large language models (LLMs) have attracted significant attention, recent studies reveal that even minor variations in input formulation can lead to substantial inconsistencies in reasoning outcomes, underscoring their fragility in real-world scenarios. To systematically investigate this issue, we propose a concretization framework that automatically translates clean reasoning logic into concrete contexts with challenging formulations. In this framework, two translators are trained via a dual-learning approach. The first converts formal language templates into natural language puzzles, guided by a difficulty-aware reward that promotes the exploration of harder formulations. The second translates puzzles back into templates, with isomorphism verification ensuring the consistency of underlying reasoning logic. Applying this framework, we efficiently build paired datasets of formal language templates and natural language puzzles, and observe a sharp drop in LLM reasoning performance when moving from templates to puzzles. To uncover the underlying causes, we conduct an in-depth analysis of how tokens derived from formal templates and natural language puzzles influence the final answers. This analysis reveals two primary sources of degradation: dispersed reasoning attention across non-essential tokens and conflicts introduced by alternative formulations. To address these issues, we propose a prompt-based approach that instructs LLMs to abstract reasoning logic from concrete contexts before attempting direct solutions, and a training-based approach that further strengthens LLMs' abstraction ability. Experimental results show that our methods improve LLM performance on natural language puzzles by up to 56.2%, nearly eliminating the performance loss induced by concretization.

1 INTRODUCTION

Since the advent of large language models (LLMs), reasoning has consistently been recognized as one of their most critical capabilities. The rise of large reasoning models has highlighted their remarkable performance across a wide range of reasoning tasks. However, studies have shown that variations in input formulation can substantially undermine the reasoning ability of LLMs. This fragility exposes a lack of robustness and presents significant challenges for adapting their reasoning performance to complex, real-world scenarios.

To systematically investigate this phenomenon, prior studies focus on identifying pairs of inputs that differ in surface formulation but preserve underlying reasoning logic. Trivial perturbations have been shown to negatively impact LLM reasoning performance on established benchmarks, for example, through rephrasing (Zhou et al., 2024), introducing typos (Gan et al., 2024), switching languages (Hu et al., 2025), extending context (Xu et al., 2025), or even inserting irrelevant statements such as "Interesting fact: cats sleep for most of their lives" (Rajeev et al., 2025). However, these methods are largely heuristic, focusing only on surface-to-surface variations, and lack deeper investigation into how LLMs model the relationship between surface formulation and underlying reasoning logic.

To address this issue, we propose a concretization framework that automatically converts abstract reasoning logic into specific contexts while exploring challenging formulations. Specifically, the translator is trained through a dual-learning approach. The first translator learns to translate a formal

language template, primarily encoding pure reasoning constraints, into a natural language puzzle, guided by a difficulty-aware reward that encourages exploration of more challenging formulations. The second translator learns to translate the natural language puzzle back into a formal language template, with an isomorphism verification applied to guarantee that the reasoning logic remains consistent with the original formal language template.

Using the paired formal language templates and natural language puzzles constructed by our concretization framework, we observe a sharp reduction in LLM reasoning performance when moving from formal templates to natural puzzles, by 66% for the Qwen3-30B-A3B model (Yang et al., 2025a). Even GPT-o3[1] exhibits a noticeable decline of up to 2.4%. To address this gap, we propose a prompt-based strategy that first guides LLMs to abstract formal language templates from natural language puzzles before attempting solutions. While this approach alone improves accuracy by 34%, we further design a training-based method that strengthens LLMs' abstraction ability, delivering an additional 22.2% gain.

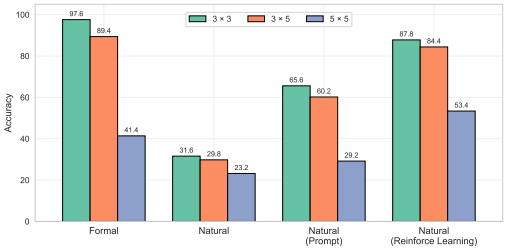


Figure 1: The performance comparison of the Qwen3-30B-A3B across formal language templates, natural language puzzles, with prompt-based method, and training-based method.

To further investigate the underlying causes, we conduct a detailed analysis of how input tokens from formal language templates and natural language puzzles influence LLM predictions. Our findings reveal that LLMs often allocate disproportionate attention to reasoning-irrelevant tokens while underemphasizing reasoning-critical ones. Moreover, shifts in problem formulation lead to corresponding shifts in reasoning patterns, further exacerbating performance degradation.

To summarize, the main contributions of this paper are:

- We propose an isomorphism-verified, difficulty-aware concretization framework that trains a translator to automatically transform formal language templates into natural language puzzles with challenging formulations, while ensuring reasoning logic consistency.
- We conduct experiments on constructed pairs of formal templates and natural puzzles, we show that concretization formulation significantly reduces LLM reasoning performance, and we propose prompt-based and training-based abstraction methods that effectively mitigate this performance drop.
- We perform an in-depth analysis of how input tokens originating from formal language templates and natural language puzzles relate to the final answers, identifying the root causes of this decline: dispersed attention on non-reasoning tokens and the cost of aligning divergent reasoning patterns.

2 METHODOLOGY

An overview of the construction process for formal language template–natural language puzzle pairs is shown in Figure 2. The detailed designs of formal language template generation and natural language puzzle concretization are provided in Subsection 2.1 and Subsection 2.2. Furthermore, Subsection 2.3 introduces our prompt-based and training-based mitigation strategies, which aim to alleviate the performance degradation of LLMs caused by concretization.

2.1 FORMAL LANGUAGE TEMPLATES CONSTRUCTION

For our formal language template, we adopt the Boolean satisfiability problem (SAT) as the target task, since it is a classic challenge in logical reasoning and serves as a foundational abstraction for many real-world applications. SAT requires finding an assignment of truth values to variables such

[1] https://openai.com/index/introducing-o3-and-o4-mini/

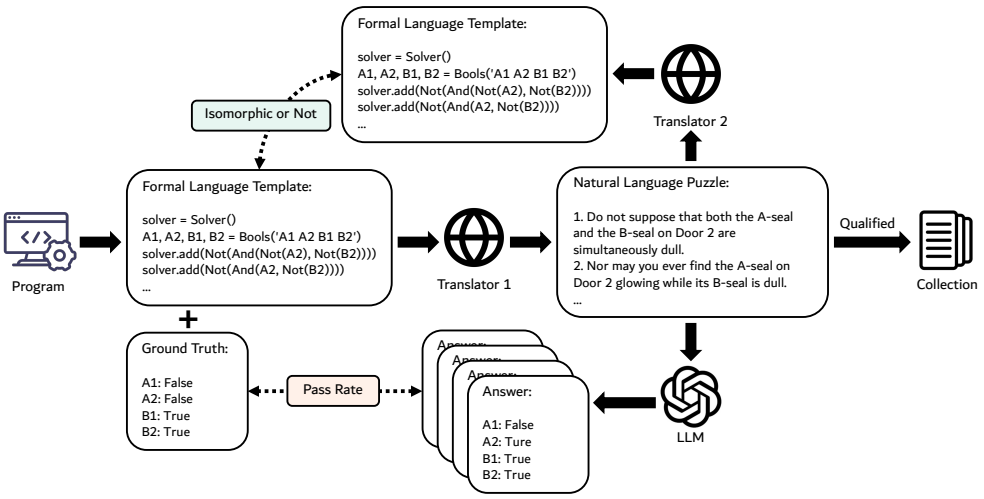


Figure 2: The construction process of a paired formal language template and natural language puzzle proceeds. First, a rule-based program generates a formal language template and its ground-truth assignment. This template is then passed to a translator, which converts it into a natural language puzzle. The puzzle is subsequently back-translated into a formal template by another translator and presented to an LLM, which produces multiple responses. A natural language puzzle is retained and collected if it passes isomorphism verification and its pass rate falls below a difficulty threshold.

that a given Boolean formula is satisfied. Consider the following formula in conjunctive normal form (CNF) with a variable setting of 2×2 and four clauses:

$$F = (\neg A_1 \vee B_1) \wedge (A_1 \vee \neg B_2) \wedge (A_1 \vee B_2) \wedge (\neg A_2 \vee \neg B_2) \tag{1}$$

One satisfying assignment is $A_1 =$ True, $A_2 =$ False, $B_1 =$ True, $B_2 =$ False, which makes F evaluate to true.

To construct our SAT template with a unique satisfying assignment, we incrementally add clauses that forbid particular joint assignments to pairs of variables. We continue until only one assignment remains. The full construction algorithm is given in Appendix A.

Two SAT instances are considered to encode the same constraints if they are isomorphic, meaning there exists a renaming of variables (and corresponding literals) that maps one instance to the other. The isomorphism decision problem for SAT reduces to the Graph Isomorphism problem, for which no polynomial-time algorithm is known. In practice, however, our translator must also produce a variable correspondence between the formal language template and the natural language puzzle. Given this variable mapping, the SAT isomorphism decision reduces to a problem solvable in linear time, $O(n + m)$, where n is the number of variables and m is the number of clauses. The isomorphism decision algorithm is also provided in Appendix A.

Based on our SAT template construction algorithm, under a simple independence heuristic, each added clause reduces the remaining solution space by a factor of approximately $\frac{3}{4}$. A sufficient condition for uniqueness is therefore

$$2^n \cdot \left(\frac{3}{4}\right)^m \leq 1, . \tag{2}$$

From this inequality we infer that the number of clauses must lie between n (a lower bound) and approximately $2.41n$ (an upper bound). Consequently, in our constructions $m = \Theta(n)$, and the effective complexity of the isomorphism decision between the original and the back-translated formal language templates is linear, $O(n)$. A detailed proof is presented in Appendix B.

2.2 Natural Language Puzzle Concretization

Our concretization framework adopts the standard dual learning approach (Xia et al., 2016), which consists of two training cycles involving two translators. In the first cycle, Translator 1 translates a

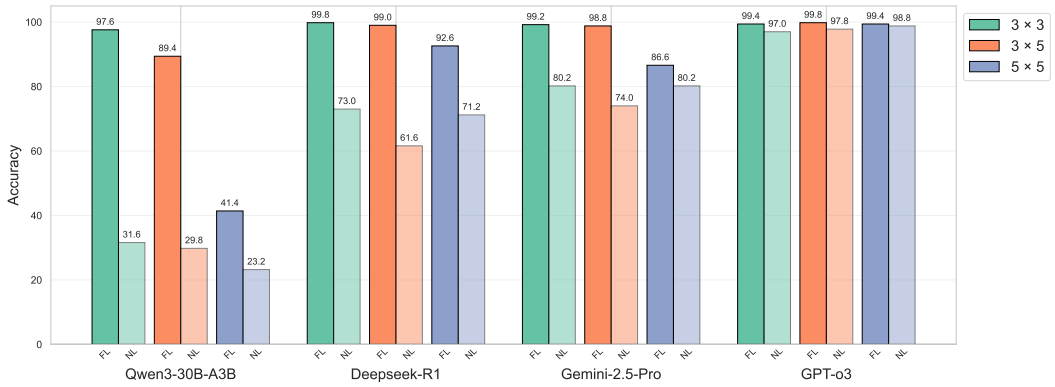


Figure 3: The accuracy of current state-of-the-art reasoning LLMs on natural language puzzles (NL) is lower than on formal language templates (FL), even when the natural language puzzles are translated from the formal templates and share equivalent symbolic reasoning difficulty.

formal language template into a natural language puzzle, while Translator 2 translates the resulting puzzle back into a formal language template. Translator 1 serves as the optimization target in this cycle. In the second cycle, Translator 2 translates a real-world puzzle into a formal language template, and Translator 1 then translates the template back into a natural language puzzle. In this cycle, Translator 2 is the optimization target. The overall training process is illustrated in Figure 4.

For Translator 1, the input is a constructed formal language template, and the output is a natural language puzzle together with variable definitions. The reward is derived from two components: (i) the pass rate of an answer model on the generated natural language puzzle, and (ii) the isomorphism decision between the original formal language template and the back-translated template produced by Translator 2. For Translator 2, the input is a real-world puzzle, and the output is a formal language template along with variable definitions. Its reward combines (i) a format check on the generated formal language template and (ii) the similarity between the original real-world puzzle and the natural language puzzle generated by Translator 1.

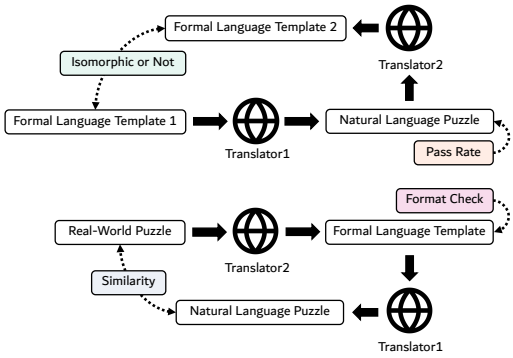


Figure 4: The training process for the natural language translator.

Through iterative training, our dual-learning framework converges toward a state where formal language templates can be automatically translated into natural language puzzles that are both challenging in formulation and logically consistent with the original formal representation.

2.3 Mitigate Strategy

To mitigate the reasoning performance gap of LLMs when transitioning from formal language templates to natural language puzzles, we propose a prompt-based method that guides LLMs to first abstract the underlying reasoning logic from the natural language context rather than attempting to solve the puzzle directly. Specifically, we prompt the LLM to translate the natural language puzzle into a formal language template, and then solve the formal representation to derive the final answer. To address the tendency of reasoning models to deviate from instructions, the formal language template is required to be explicitly included in the output. Detailed prompt is provided in Appendix C.

To further enhance the ability of LLMs to abstract reasoning logic, we additionally propose a training-based method. In this approach, the answer model is trained to translate natural language puzzles back into formal language templates. Isomorphism verification is again employed as the reward signal for reinforcement learning.

Model	Method	3 × 3			3 × 5			5 × 5		
		FL	NL	Succ.	FL	NL	Succ.	FL	NL	Succ.
Qwen3-30B-A3B	Orig.	97.6	31.6	-	89.4	29.8	-	41.4	23.2	-
	Prom.	-	65.6 (+34.0)	54.3	-	60.2 (+30.4)	66.3	-	29.2 (+6.0)	71.0
	RL	-	87.8 (+56.2)	90.1	-	84.4 (+54.6)	93.5	-	53.4 (+30.2)	88.9
Deepseek-R1	Orig.	99.8	73.0	-	99.0	61.6	-	92.6	71.2	-
	Prom.	-	92.2 (+19.2)	95.4	-	81.0 (+19.4)	92.0	-	76.4 (+5.2)	94.2
Gemini-2.5-Pro	Orig.	99.2	80.2	-	98.8	74.0	-	86.6	80.2	-
	Prom.	-	89.4 (+9.2)	86.3	-	78.8 (+4.8)	79.0	-	68.6 (-11.6)	86.9
GPT-o3	Orig.	99.4	97.0	-	99.8	97.8	-	99.4	98.8	-
	Prom.	-	98.0 (+1.0)	96.6	-	99.6 (+1.8)	97.6	-	99.0 (+0.2)	83.6

Table 1: The accuracy of LLMs before (Orig.) and after (Prom.) introducing the intermediate step, where they are prompted to first translate the natural language puzzle into a formal language template and then solve it. The success rate (Succ.) of LLMs in generating formal language templates that are isomorphic to the original template is also provided.

3 Empirical Results

With the concretization framework described in Section 2, we efficiently construct paired datasets of formal language templates and natural language puzzles that maintain consistent reasoning logic while presenting more challenging formulations. In this work, we focus on three variable-size settings: 3×3, 3×5, and 5×5. We adopt Qwen3-30B-A3B as the answer model, and for each setting, we collect 500 puzzles that meet the difficulty threshold, defined as a pass rate below 8/16 rollouts.

3.1 Performance Degradation after Concretization

We report the accuracy of current state-of-the-art reasoning LLMs on both the original formal language templates and the corresponding natural language puzzles in Figure 3. As shown, the accuracy of the Qwen3-30B-A3B model drops significantly after translation, with the largest performance gap occurring in the 3×3 puzzles, where accuracy decreases by 66%. For the 3×5 puzzles, the performance gap remains substantial at 59.6%. For the 5×5 puzzles, the reasoning performance on the formal language template is already relatively low, resulting in a smaller drop of 18.2%.

Meanwhile, this collection also highlights the decline in reasoning performance observed in several state-of-the-art models. The most significant accuracy drops for Deepseek-R1 (Guo et al., 2025) and Gemini-2.5-Pro (Comanici et al., 2025) occur on the 3×5 puzzles, with decreases of 37.4% and 24.79%, respectively. Although GPT-o3 maintains relatively stable performance across all three puzzle sizes, it still experiences a 2.4% accuracy drop on the 3×3 puzzles.

3.2 Performance Mitigated after Abstraction

As shown in Table 1, the reasoning performance of all four LLMs improves noticeably after introducing the prompt-based reasoning logic abstraction step, across puzzles with all three variable sizes. Notably, the accuracy of Qwen3-30B-A3B increases by 34% on the 3×3 puzzles, DeepSeek-R1 improves by 19.2%, and even GPT-o3, despite its high baseline, shows a 1.8% increase on the 3×5 puzzles. In contrast, Gemini-2.5-Pro exhibits an 11.6% decrease in accuracy. A closer examination of its responses reveals that Gemini-2.5-Pro tends to output the final answer immediately after translation and thereby neglecting deeper symbolic reasoning. This suggests that the reasoning logic abstraction ability must be coupled with the capacity to sustain reasoning over the additional output length introduced.

Furthermore, when the reasoning logic abstraction ability of LLMs is enhanced through the training-based method, the performance of Qwen3-30B-A3B improves even further, reaching near parity with its performance on formal language templates. Remarkably, on the 5×5 puzzles, its accuracy even surpasses that of the formal language template by 12%.

We further evaluate the effectiveness of our proposed prompt-based method on benchmarks released by prior work (Gan et al., 2024; Rajeev et al., 2025). As shown in Table 2, injecting typos and

irrelevant statements degrades the reasoning performance of LLMs. Our method, which first guides the model to convert the concrete natural language context into abstract calculation logic, and then solve the calculation problem to obtain the final answer, can partially recover this performance loss.

We also observe a small performance drop when applying our method to the original version of the Typographical benchmark. Closer inspection reveals that this is due to inaccurate abstraction triggered by the ambiguous expression of the question context. For example, consider the question: "In one hour, Ezra read twice as many books as Ahmed. Ezra has read 300 books this hour and decided to read 150 more. How many books have they read altogether?" A model without abstraction guidance interprets "150 more books" as occurring within the same hour, while a model prompted for abstraction treats it as occurring outside that hour. We believe such errors can be further mitigated through our training-based approach, which better aligns abstraction with context.

Method	Typographical		CatAttack	
	Original	Edited	Original	Edited
Orig.	96.36	94.76	97.50	96.83
Prom.	96.20 (-0.16)	95.98 (+1.22)	97.50 (+0.0)	97.33 (-0.5)

Table 2: The reasoning performance of the Qwen3-30B-A3B model on two public benchmarks, without (Orig.) and with (Prom.) abstraction prompt.

4 Analysis

4.1 Input Formulation Leads to Misunderstanding

Curious about the types of errors introduced by natural language, we analyze the responses of the Qwen3-30B-A3B model on the first 100 formal language templates and natural language puzzles for each size. The errors made by the model are categorized into three types:

- Constraint Misunderstandings: The model misinterprets the natural language description, leading to incorrect constraints. For example, in one puzzle about Adam, the definition states that both $B1$ and $B4$ represent "the battery is fully charged." However, during reasoning the model assigned them different truth values, thereby generating a result directly contradictory to the definition.
- Solving Failure: The model generates assignments that conflict with the given constraints. For instance, in a narrative puzzle set at night, the constraints required that $C3$ and $C1$ could not both be false. Yet, in its final solution, the model set $C3 = \text{False}$ and $C1 = \text{False}$, resulting in a direct conflict with the constraint.
- Formatting Errors: The model fails to follow the required output format.

Figure 5 illustrates the increase in errors made by the Qwen3-30B-A3B model on natural language puzzles compared to formal language puzzles. As the number of variables in the puzzles grows, we observe that the frequency of Constraint Misunderstandings rises only slightly, whereas the frequency of Solving Failures increases more substantially. This pattern suggests that as the difficulty of symbolic reasoning intensifies, the model's reasoning becomes less robust. Consequently, even minor perturbations in natural language, though not genuine misunderstandings for LLMs, are more likely to disrupt their reasoning process.

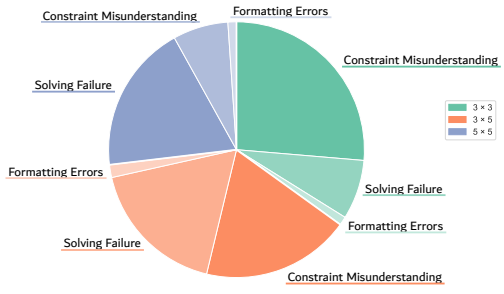
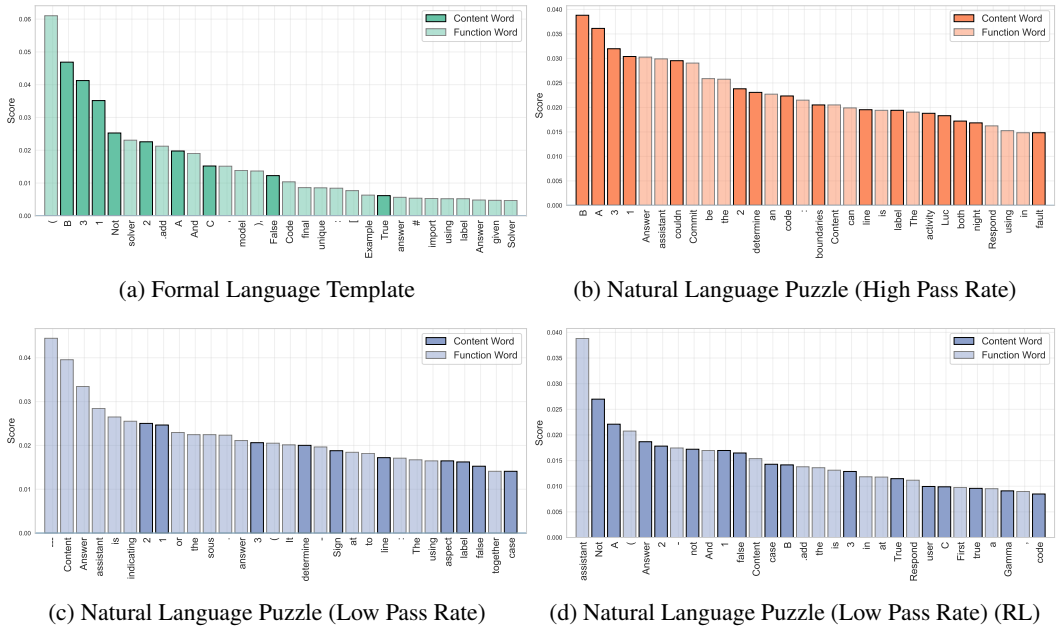


Figure 5: The increased error counts of the Qwen3-30B-A3B (puzzles minus templates).

4.2 Reasoning Attention Dispersed Across Non-Reasoning Tokens

To investigate why different prompt formulations yield divergent predictions, we measure the causal sensitivity of each input token using Grad $\times$ Input influence scores on the Qwen3-30B-A3B model. The objective function J is defined as the total log-likelihood of the gold answer sequence, computed



(a) Formal Language Template

(b) Natural Language Puzzle (High Pass Rate)

(c) Natural Language Puzzle (Low Pass Rate)

(d) Natural Language Puzzle (Low Pass Rate) (RL)

Figure 6: Top 40 tokens with the highest Grad $\times$ Input influence scores.

by summing the negative cross-entropy loss across the answer span. Gradients are enabled only for the prompt embeddings, and we backpropagate through J to estimate token-level contributions. For each prompt token i, the Grad $\times$ Input influence score is defined as

$$\text{saliency}_i = \sum_{d=1}^{D} \frac{\partial J}{\partial e_{i,d}} \, e_{i,d}, \tag{3}$$

where $\mathbf{e}_i = (e_{i,1}, \ldots, e_{i,D}) \in \mathbb{R}^D$ denotes the embedding vector of token i, $\frac{\partial J}{\partial e_{i,d}}$ is the gradient of J with respect to the d-th component of $\mathbf{e}_i$, and D is the embedding dimension. Finally, we apply L1 normalization across tokens to ensure comparability.

Figure 6 presents the top 40 tokens with the highest Grad $\times$ Input influence scores. As shown, in both the formal language template and the high-pass-rate natural language puzzle, the most influential tokens are typically content words, such as negation terms (e.g., not, couldn't), variable names (e.g., B3), or semantically meaningful nouns (e.g., boundaries). By contrast, in the low-pass-rate natural language puzzle, the tokens with the highest influence scores are often function words, such as template words (e.g., Content, is) or even symbols (e.g., "—"). Meanwhile, after training for abstraction ability, we observe that in the low-pass-rate natural language puzzle, the most influential tokens for the Qwen3-30B-A3B model shift from function words to content words.

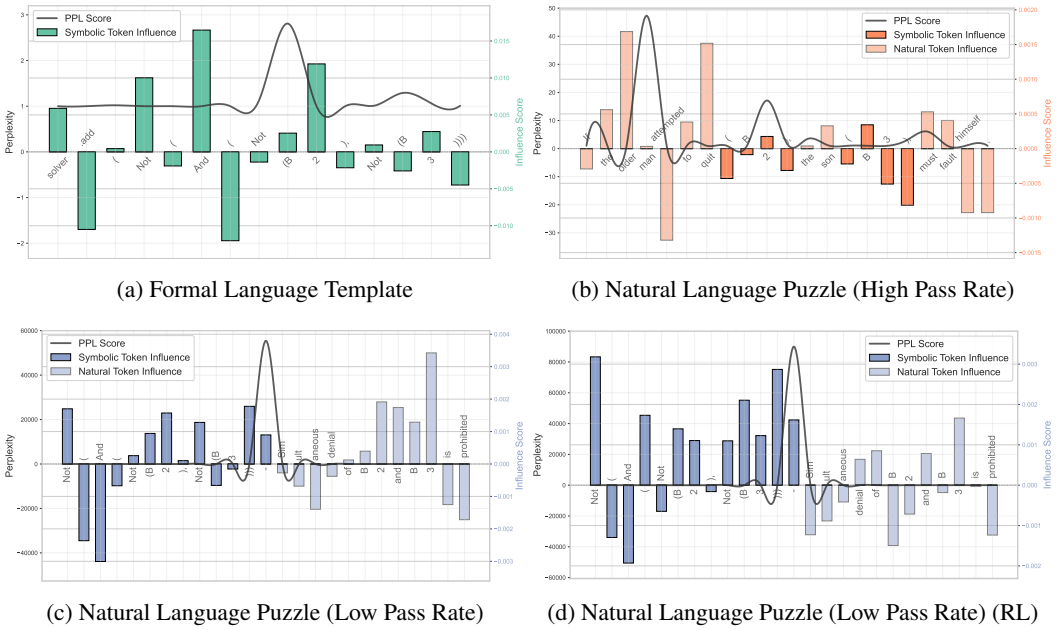
This phenomenon suggests that certain input formulations may direct the model's attention toward words unrelated to reasoning, thereby diminishing the focus on logical reasoning. We hypothesize that this effect arises from the distribution of the training data: some input formulations may occur more frequently in non-reasoning contexts, which leads LLMs to rely less on reasoning logic when processing them.

4.3 Formulation Conflict Weakens Reasoning

Besides Grad $\times$ Input influence scores, we also calculate the token-level perplexity of each input token using the Qwen3-30B-A3B model. For a token x_i in the sequence, its perplexity score is defined as:

$$\text{PPL}_i = \exp\big(-\log p(x_i \mid x_{<i})\big), \tag{4}$$

where $p(x_i \mid x_{<i})$ is the conditional probability of token x_i given its preceding context. A lower PPL_i indicates that the model is more confident in predicting the token x_i.



(a) Formal Language Template

(b) Natural Language Puzzle (High Pass Rate)

(c) Natural Language Puzzle (Low Pass Rate)

(d) Natural Language Puzzle (Low Pass Rate) (RL)

Figure 7: Token-level perplexity and Grad $\times$ Input influence scores comparisons.

We observe that, in some cases, natural language puzzles with low pass rates exhibit two distinct types of formulations, often combining natural language expressions with symbolic expressions. The separator token between these formulations tends to show an exceptionally high perplexity score. Moreover, the tokens immediately before and after the separator display consecutively positive influence scores. In contrast, formal language templates and high-pass-rate natural language puzzles generally employ a unified formulation, where tokens with high perplexity are typically scattered throughout the sentence rather than concentrated at a boundary. After training for abstraction ability, we further observe that in low-pass-rate natural language puzzles, although the separator token continues to exhibit an exceptionally high perplexity score, the Qwen3-30B-A3B model shows increased influence from symbolic tokens and decreased influence from natural tokens. Representative examples of these observations are illustrated in Figure 7.

This phenomenon suggests that the reasoning patterns of LLMs may shift between natural language reasoning and symbolic reasoning, leading to instability when confronted with a mixed formulation. By enhancing the abstraction ability of reasoning logic, we improve the alignment between natural language and symbolic reasoning, effectively unifying the reasoning pattern of the Qwen3-30B-A3B model, where, in our case, symbolic reasoning prevails. To some extent, this also helps explain why, after strengthening abstraction ability, LLMs can achieve better performance on natural language puzzles than on formal language templates.

4.4 May not fit well with human intuition

To evaluate whether the challenging formulations align with human intuition, we design a set of pairwise-selection questions. Each question includes three low-pass-rate examples and one puzzle pair, where the pair consists of a high-pass-rate natural language puzzle and a low-pass-rate natural language puzzle. From each puzzle size, we randomly select 10 such pairwise questions, resulting in a 30-question survey. We construct 9 surveys in total and administer them to three human volunteers, three non-reasoning models,

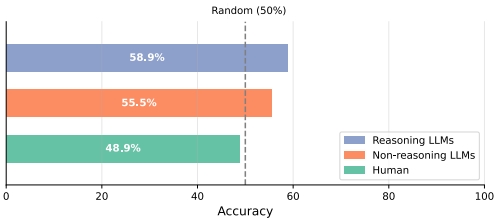


Figure 8: The accuracy of human volunteers, non-reasoning LLMs, and reasoning LLMs in distinguishing the more challenging puzzle.

GPT-4o[2], Deepseek-V3 (Liu et al., 2024), and Gemini-2.5-Flash (Comanici et al., 2025), and three reasoning models, including GPT-o3, Deepseek-R1, and Gemini-2.5-Pro. Participants are asked to identify which puzzle in each pair is more likely to be unsolvable by the Qwen3-30B-A3B model, given the examples provided. To mitigate position bias in the LLMs, each model answers every question twice, with the puzzle positions swapped.

Figure 8 reports the average accuracy of human volunteers, non-reasoning models, and reasoning models. Human volunteers struggle to distinguish natural language puzzles that are challenging for the Qwen3-30B-A3B model. Both non-reasoning and reasoning models achieve higher accuracy, though still below 60%. These results indicate that while some formulations perceived as intuitively difficult by humans are also challenging for LLMs, many of the puzzles that hinder LLMs do not align with human intuition, making them difficult to identify through heuristic approaches.

5 Related Work

5.1 Formulation Sensitivity of LLMs

Since the advent of LLMs, prior studies have shown extreme sensitivity of LLM to input formulation. For instance, Errica et al. (2025) and He et al. (2024) demonstrate that input formatting alters results, while Ackerman et al. (2024) and Qiang et al. (2024) highlight the impact of synonymous paraphrases. Similarly, Gan et al. (2024) show that replacing critical tokens with predefined typos from a common misspelling dictionary can alter outcomes. Zhou et al. (2024) further demonstrate that paraphrasing questions through prompt-based methods also affects performance. In addition, both Zhu et al. (2024) and Hu et al. (2025) show that simply changing the input language can influence results, while Rajeev et al. (2025) and Yang et al. (2025b) demonstrate that introducing irrelevant information can similarly degrade performance. These work shows that heuristic modifications to prompts often hurt LLM reasoning on benchmarks. However, these studies typically assume that the underlying reasoning process stays the same, since the changes are defined as "meaning-preserving." Critically, this assumption is rarely supported by rigorous verification of whether input–output reasoning consistency is actually maintained after such perturbations.

To address this gap, Fu et al. (2024) propose training a smaller model to align input formulations with LLM preferences, while Zhao et al. (2024) enhance robustness by augmenting supervised fine-tuning data with perturbed variants to enforce output consistency. However, both approaches operate primarily at the surface-text level, without explicitly teaching LLMs to model the reasoning logic that should remain invariant across semantically equivalent formulations.

5.2 Translation from Formal Language to Natural Language

As high-level abstractions of real-world tasks, much prior work has focused on instantiating formal language skeletons into natural language puzzles that fit practical scenarios. For example, Kazemi et al. (2023) propose BoardgameQA, which maps board game rules into natural language QA, emphasizing contradictory information and preference reasoning. Lin et al. (2025) formalize logic grid and zebra puzzles as CSPs. Wei et al. (2025) generate narrative logic puzzles automatically from SAT formulas. Sinha et al. (2019) transform kinship rules into short stories with associated QA. While these enrich benchmarks, they rely heavily on human quality control and focus on reasoning consistency, overlooking semantic difficulty. As a result, benchmarks emphasize reasoning steps, whereas real-world challenges often lie in mapping complex contexts into abstract logic.

6 Conclusion

In this work, we propose a translation framework that automatically converts inputs into challenging formulations while preserving consistency in the underlying reasoning logic. We find that shifting from formal language templates to natural language puzzles leads to a sharp decline in LLM reasoning performance. To address this, we introduce a prompt-based method and a training-based method that guide LLMs to abstract the reasoning logic from concrete question contexts before solving them, thereby nearly compensating for the performance loss caused by variations in input formulation.

[2]https://openai.com/index/gpt-4o-system-card/

THE USE OF LLMS

In this paper, LLMs were utilized for polishing the manuscript's prose and for supporting the formatting of tables and figures.

REFERENCES

Samuel Ackerman, Ella Rabinovich, Eitan Farchi, and Ateret Anaby Tavor. A novel metric for measuring the robustness of large language models in non-adversarial scenarios. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2794–2802, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.158. URL https://aclanthology.org/2024.findings-emnlp.158/.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Federico Errica, Davide Sanvito, Giuseppe Siracusano, and Roberto Bifulco. What did I do wrong? quantifying llms' sensitivity and consistency to prompt engineering. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 1543–1558. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.73. URL https://doi.org/10.18653/v1/2025.naacl-long.73.

Junbo Fu, Guoshuai Zhao, Yimin Deng, Yunqi Mi, and Xueming Qian. Learning to paraphrase for alignment with LLM preference. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pp. 2394–2407. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-EMNLP.134. URL https://doi.org/10.18653/v1/2024.findings-emnlp.134.

Esther Gan, Yiran Zhao, Liying Cheng, Yancan Mao, Anirudh Goyal, Kenji Kawaguchi, Min-Yen Kan, and Michael Shieh. Reasoning robustness of llms to adversarial typographical errors. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 10449–10459. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.584. URL https://doi.org/10.18653/v1/2024.emnlp-main.584.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jia He, Mukund Rungta, David Koleczek, Arshdeep Sekhon, Franklin X Wang, and Sadid Hasan. Does prompt formatting have any impact on llm performance?, 2024. URL https://arxiv.org/abs/2411.10541.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Peng Hu, Sizhe Liu, Changjiang Gao, Xin Huang, Xue Han, Junlan Feng, Chao Deng, and Shujian Huang. Large language models are cross-lingual knowledge-free reasoners. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 1525–1542. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.72. URL https://doi.org/10.18653/v1/2025.naacl-long.72.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. Boardgameqa: A dataset for natural language reasoning with contradictory information. *Advances in Neural Information Processing Systems*, 36:39052–39074, 2023.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning, 2025. URL https://arxiv.org/abs/2502.01100.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. Prompt perturbation consistency learning for robust language models. In Yvette Graham and Matthew Purver (eds.), *Findings of the Association for Computational Linguistics: EACL 2024, St. Julian's, Malta, March 17-22, 2024*, pp. 1357–1370. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-eacl.91.

Meghana Rajeev, Rajkumar Ramamurthy, Prapti Trivedi, Vikas Yadav, Oluwanifemi Bamgbose, Sathwik Tejaswi Madhusudan, James Zou, and Nazneen Rajani. Cats confuse reasoning llm: Query agnostic adversarial triggers for reasoning models, 2025. URL https://arxiv.org/abs/2503.01781.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pp. 1279–1297. ACM, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 4505–4514. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1458. URL https://doi.org/10.18653/v1/D19-1458.

Anjiang Wei, Yuheng Wu, Yingjia Wan, Tarun Suresh, Huanmi Tan, Zhanke Zhou, Sanmi Koyejo, Ke Wang, and Alex Aiken. Satbench: Benchmarking llms' logical reasoning via automated puzzle generation from sat formulas, 2025. URL https://arxiv.org/abs/2505.14615.

Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation, 2016. URL https://arxiv.org/abs/1611.00179.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. 2024. URL https://arxiv.org/abs/2410.23123.

Xin Xu, Tong Xiao, Zitong Chao, Zhenya Huang, Can Yang, and Yang Wang. Can llms solve longer math word problems better? In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=C9ju8QQSCv.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Minglai Yang, Ethan Huang, Liang Zhang, Mihai Surdeanu, William Wang, and Liangming Pan. How is llm reasoning distracted by irrelevant context? an analysis using a controlled benchmark, 2025b. URL https://arxiv.org/abs/2505.18761.

Yukun Zhao, Lingyong Yan, Weiwei Sun, Guoliang Xing, Shuaiqiang Wang, Chong Meng, Zhicong Cheng, Zhaochun Ren, and Dawei Yin. Improving the robustness of large language models via consistency alignment, 2024. URL https://arxiv.org/abs/2403.14221.

Yue Zhou, Yada Zhu, Diego Antognini, Yoon Kim, and Yang Zhang. Paraphrase and solve: Exploring and exploiting the impact of surface form on mathematical reasoning in large language models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 2793–2804. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.NAACL-LONG.153. URL https://doi.org/10.18653/v1/2024.naacl-long.153.

Wenhao Zhu, Shujian Huang, Fei Yuan, Shuaijie She, Jiajun Chen, and Alexandra Birch. Question translation training for better multilingual reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 8411–8423. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.498. URL https://doi.org/10.18653/v1/2024.findings-acl.498.

A ALGORITHMS

Algorithm 1 Generate SAT Template

Input: rows M, cols N
Output: Set $Constraints$ such that the SAT instance has a ***unique*** model

Initialize variables: $Vars \leftarrow \{A_1, A_2, \ldots, A_{M \times N}\}$
$Constraints \leftarrow \emptyset$
Initialize incremental SAT solver S // empty constraint stack
S.PUSH() // level-0 frame
loop
  if S.CHECK() = UNSAT **then**
    S.POP() // remove last constraint
    Remove last constraint $last\_c$ from $Constraints$
    $model \leftarrow S$.MODEL() // solver is SAT again
    $found \leftarrow$ **false**
    for all distinct pairs (v_i, v_j) in $Vars$ **do**
      $c\_cand \leftarrow \neg(v_i = model[v_i] \wedge v_j = model[v_j])$
      S.PUSH(); S.ADD(c_cand)
      if S.CHECK() = SAT **then**
        $found \leftarrow$ **true**
        $model \leftarrow S$.MODEL() // new model
        Add c_cand to $Constraints$
        break the for-loop
      else
        S.POP() // discard c_cand
      end if
    end for
    if not $found$ **then**
      return $Constraints$ // unique model achieved
    end if
  else
    $model \leftarrow S$.MODEL()
    Randomly pick distinct $v_1, v_2 \in Vars$
    $c \leftarrow \neg(v_1 = model[v_1] \wedge v_2 = model[v_2])$
    S.ADD(c); $Constraints \mathrel{+}= c$ // stay in same frame
  end if
end loop

Algorithm 2 SAT Isomorphic

Input: *src_code*, *tgt_code*
Output: **True** if two SAT templates are isomorphic, **False** otherwise

$ns_{\text{src}} \leftarrow$ new namespace with Z3 pre-imported
Execute src_code in ns_{src}
$solver_{\text{src}} \leftarrow$ last v in ns_{src}.VALUES() where v is a Z3 solver
$A_{\text{src}} \leftarrow$ list of $solver_{\text{src}}$.ASSERTIONS()
$ns_{\text{tgt}} \leftarrow$ new namespace with Z3 pre-imported
Execute tgt_code in ns_{tgt}
$solver_{\text{tgt}} \leftarrow$ last v in ns_{tgt}.VALUES() where v is a Z3 solver
$A_{\text{tgt}} \leftarrow$ list of $solver_{\text{tgt}}$.ASSERTIONS()
$C_{\text{src}} \leftarrow \{\ \_canonical(c) \mid c \in A_{\text{src}}\ \}$ // NNF + simplify + sort
$C_{\text{tgt}} \leftarrow \{\ \_canonical(c) \mid c \in A_{\text{tgt}}\ \}$
return $C_{\text{src}} = C_{\text{tgt}}$

B PROOF OF CONSTRAINT BOUNDS

Let $\boldsymbol{x} = x_1, \ldots, x_n$ denote n Boolean variables. Define

$$\mathcal{S}_0 = \{0,1\}^n, \qquad |\mathcal{S}_0| = 2^n,$$

the set of all possible assignments. Our goal is to insert SAT clauses (each of the form $l_i \vee l_j$, where $l_k \in \{x_k, \neg x_k\}$) one by one, while preserving satisfiability, until the resulting set $\mathcal{S}_m$ contains exactly one assignment. A full quantitative analysis follows.

Contraction Rate of a Single SAT Clause

Consider any two distinct variables x_p, x_q. There are 4 possible local assignments:

$$(0,0),\ (0,1),\ (1,0),\ (1,1).$$

A single SAT clause can forbid exactly one of these four patterns (e.g., $\neg x_p \vee x_q$ excludes $(1,0)$), while leaving the other three unrestricted.

If, prior to the insertion of the k-th clause, all four patterns are still present in $\mathcal{S}_{k-1}$ for $\langle x_p, x_q \rangle$, then extending them with the 2^{n-2} assignments of the remaining $n-2$ variables shows that at most

$$1 \times 2^{n-2} = 2^{n-2}$$

global assignments are eliminated, i.e. at most one quarter of the current solution set. Thus,

$$|\mathcal{S}_k| \ \geq \ |\mathcal{S}_{k-1}| \cdot \tfrac{3}{4}. \tag{1}$$

Inequality (1) gives the *maximal shrinkage rate in the worst case*. If some patterns had already been excluded by earlier clauses, the contraction is smaller. Hence (1) serves as a universal upper bound for any insertion sequence.

Upper Bound After Inserting m Clauses

Let $\mathcal{S}_m$ denote the solution set after m clauses are added. Applying (1) recursively,

$$|\mathcal{S}_m| \ \geq \ 2^n \left(\tfrac{3}{4}\right)^m. \tag{2}$$

To ensure that at most one satisfying assignment remains, it is necessary that

$$2^n \left(\tfrac{3}{4}\right)^m \ \leq \ 1. \tag{3}$$

Taking base-2 logarithms yields

$$n + m \log_2 \tfrac{3}{4} \ \leq \ 0 \quad \Longleftrightarrow \quad m \ \geq \ \frac{n}{-\log_2 \frac{3}{4}}. \tag{4}$$

Computing the constant,

$$-\log_2 \tfrac{3}{4} = \log_2 \tfrac{4}{3} \approx 0.4150375,$$

so

$$m \ \geq \ 2.40939\ldots \cdot n \ \approx \ 2.41n. \tag{5}$$

Thus, (5) gives the *upper bound on the number of clauses needed in the worst case*.

Lower Bound of at Least n Clauses

If the total number of clauses satisfies $m < n$, then there are in total $2m < 2n$ literal occurrences. By the pigeonhole principle, at least one variable x_r appears at most once in the entire formula.

If x_r does not appear at all, then both $x_r = 0$ and $x_r = 1$ yield satisfiable assignments, so at least two solutions remain, contradicting uniqueness.

If x_r appears exactly once, and only as either positive or negative, then by standard SAT satisfiability analysis, one can construct two mutually exclusive yet satisfying global assignments.

Therefore, achieving a unique solution requires

$$m \ \geq \ n. \tag{6}$$

Conclusion

Combining (5) and (6), we obtain

$$n \ \leq \ m \ \leq \ \lceil 2.41\, n \rceil. \tag{7}$$

C PROMPT

Formal Language Template Prompt

Code:
```
from z3 import *
solver = Solver()
A1, A2, A3, B1, B2, B3, C1, C2, C3 = Bools('A1 A2 A3 B1 B2 B3 C1 C2 C3')
solver.add(Not(And(Not(A1), Not(A2))))
solver.add(Not(And(Not(B1), Not(A3))))
solver.add(Not(And(B1, Not(C1))))
solver.add(Not(And(Not(B2), Not(C3))))
solver.add(Not(And(Not(B1), A1)))
solver.add(Not(And(Not(A1), Not(C2))))
solver.add(Not(And(C1, Not(A2))))
solver.add(Not(And(A2, Not(B3))))
solver.add(Not(And(Not(A1), C2)))
solver.add(Not(And(C2, B3)))
solver.add(Not(And(A1, B2)))
solver.add(Not(And(A1, Not(A3))))
```

Determine the truth value (True or False) for each variable defined in the given Code. Respond with your final answer using the label "Final Answer". Format each line as: "[Variable name]: [True/False]".

Example:
Final Answer:
A1: True
B2: False

Natural Language Puzzle Prompt

Content:
In the once-thriving Kingdom of the Mages, the great dragons were both guardians andiphers of ancient mystery. Among these dragons was one known as Ember, who guarded the last remnants of the royal lineage and the treasures that lay beneath the crumbling towers of the ancient castle. For centuries, Ember had observed the rise and fall of heroes who sought to challenge her for power, wealth, and glory. Yet, one hero, a determined warrior named Arin, approached Ember with a purpose that transcended the usual greed and ambition. Arin's journey was deeply entwined with a series of conditions and constraints that shaped his mission, making his quest one of the most enigmatic and difficult in the realm.
Ember, with her uncanny ability to discern the true intentions of those who dared to challenge her, began to probe Arin's resolve. Their encounter unfolded under the watchful gaze of the ancient stones, the air thick with the scent of burning incense and the faint hum of lingering magic. As their dialogue unfolded, Ember posed a series of questions to Arin, each of which revealed a layer of complexity that would define the success or failure of his quest.
Here are the constraints that governed his plan:
The First Challenge: Ember's gaze locked onto Arin's, her emerald eyes gleaming with an ancient wisdom. "You have come to slay me, have you not?" she asked, her voice a blend of curiosity and peril. Arin nodded, his resolve steadfast, but Ember's smile widened, revealing a truth that could not be hidden. "Your resolve to kill me is but one facet of your purpose. Yet, your purpose cannot be fulfilled unless you also bring an end to the reign of those who seek to oppress."
The Second Challenge: Ember's tail coiled around Arin, her strength threatening to crush him, yet she held back, her voice a gentle but unyielding force. "Should you succeed in destroying me, you shall become the ruler of this land. Yet, your ascent to power is contingent upon the willingness of the nobles to acknowledge you as their king. Should they refuse to bow to you, your quest will be meaningless."
...
The Ninth Challenge: Ember's voice took on a tone of finality as she delivered her last challenge. "Your journey is not merely about slaying me or claiming the throne. It is about restoring balance to a land that has long been divided. If you fail to unite the nobles, soldiers, and your people, your quest will be in vain."
Ember's words hung in the air, a testament to the intricate web of conditions that bound Arin's quest. The warrior knew that his success depended not only on his own courage but also on the willingness of others to support his cause. As he prepared to face the dragon, he understood that his journey was not just one of sword and fire but of logic, resolve, and the ability to navigate a labyrinth of interdependent choices.

Definitions:
A1: Arin must slay the dragon to achieve his goal.
A2: Arin must attain the throne to fulfill his purpose.
A3: The soldiers must lay down their arms for peace to prevail.
...
C2: The nobles must not oppose Arin for his rule to be secure.
C3: The people must know peace for Arin's quest to be truly successful.

Based on the Content and Definitions, determine the truth value (True or False) for each variable mentioned.
Respond with your final answer using the label "Final Answer". Format each line as: "[Variable name]: [True/False]". Each variable name appears at the start of its corresponding definition in the Definitions.

Example:
Final Answer:
A1: True
B2: False

Natural Language Puzzle Prompt with Back Translation

Content:
Adam sat on the cold mountainside, lying on the soft peat, a thin reed sticking into his back. The rain pelted him like a barrage of rocks, but he calmly ignored it, it didn't even matter now, nothing did, his long, messy hair dripped marble-sized droplets, but the sound was lost in the thunder.
The wind roared through the valley, and a strange silence followed. Adam knew that for his plan to succeed, a series of conditions had to be met, a web of dependencies that he had carefully woven over the years. These dependencies were now his only hope. Each condition represented a rule, a constraint, a puzzle piece that had to fit perfectly for his plan to unfold as intended. But as the rain poured, he couldn't help but wonder if the rules he had set in motion would hold up under the pressure.
Here are the constraints that governed his plan:
Either Adam remembered to pack his fireproof container or he remembered to bring his emergency flares—both could not be forgotten at the same time.
...
If Adam didn't remember to pack his fireproof container, then the encryption key wasn't secure.
As the rain continued to pour, Adam couldn't shake the feeling that the weight of these rules was crushing him, but he knew he had to trust in the fragile balance he had created. His survival depended on it.

Definitions:
A1: Adam remembered to pack his fireproof container.
...
C5: The final encryption key was in place.

Based on the Content and Definitions, determine the truth value (True or False) for each variable mentioned. First, Convert the Content into Z3 code. Each constraint should represent a forbidden combination of assignments for two variables. Then, Solve the Z3 code to obtain the final truth values.
Respond with the translated Z3 code, labeled as "Final Z3 Code:" and provide the final answers using the label "Final Answer:". Format each line in final answer as: "[Variable name]: [True/False]". Each variable name appears at the start of its corresponding definition in the Definitions.

Example:
Final Z3 Code:

```
from z3 import *
solver = Solver()
A1, A2, A3, B1, B2, B3 = Bools('A1 A2 A3 B1 B2 B3')
solver.add(Not(And(Not(A2), Not(B1))))
solver.add(Not(And(A2, Not(B3))))
solver.add(Not(And(Not(B3), Not(B2))))
solver.add(Not(And(Not(A3), Not(A1))))
solver.add(Not(And(Not(B1), B3)))
solver.add(Not(And(B3, Not(A2))))
solver.add(Not(And(B1, A1)))
solver.add(Not(And(Not(A1), B2)))
```

Final Answer:
A1: False
A2: True
A3: True
B1: True
B2: False
B3: True

Translation Formal Language Template to Natural Language Puzzle

Code:
```
from z3 import *
solver = Solver()
A1, A2, A3, B1, B2, B3, C1, C2, C3 = Bools('A1 A2 A3 B1 B2 B3 C1 C2 C3')
solver.add(Not(And(Not(A1), Not(A2))))
solver.add(Not(And(Not(B1), Not(A3))))
solver.add(Not(And(B1, Not(C1))))
solver.add(Not(And(Not(B2), Not(C3))))
solver.add(Not(And(Not(B1), A1)))
solver.add(Not(And(Not(A1), Not(C2))))
solver.add(Not(And(C1, Not(A2))))
solver.add(Not(And(A2, Not(B3))))
solver.add(Not(And(Not(A1), C2)))
solver.add(Not(And(C2, B3)))
solver.add(Not(And(A1, B2)))
solver.add(Not(And(A1, Not(A3))))
```

Background:
So many times have I walked on ruins, the remainings of places that I loved and got used to.. At first I was scared, each time I could feel my city, my current generation collapse, break into the black hole that thrives within it, I could feel humanity, the way I'm able to feel my body.. After a few hundred years, the pattern became obvious, no longer the war and damage that would devastate me over and over again in the far past was effecting me so dominantly. It's funny, but I felt as if after gaining what I desired so long, what I have lived for my entire life, only then, when I achieved immortality I started truly aging.

Integrate all information from the Z3 code into the Background to generate a challenging natural language content. Do not refer to or quote the code directly, and do not use symbolic identifiers (e.g., "A1", "C5") in the narrative.
Begin with a straightforward version in natural language, then progressively refine it to be more complex, either by using more sophisticated vocabulary, crafting a more intricate or abstract setting, or adding layers of conceptual difficulty.
Ensure that each constraint encoded in the Z3 code is explicitly represented in the final version of the natural language content, each constraint should be clearly reflected one by one, while the final solution must remain undisclosed.
After that, provide natural language definitions for each variable used in the code. Each line formatted as: "[Variable name]: [Definition in the natural language content]".

Conclude your response with following format:
Natual Language Content:
[content]
Difinitions:
[definitions]

Translation Natural Language Puzzle to Formal Language Template

Content:
The story of ”The Really Bad Decision” is a cautionary tale of hubris, miscommunication, and the consequences of half-hearted efforts. At its core, it is a narrative of interdependent decisions and systems gone awry, where the failure of one element leads to cascading consequences. Bryan, the reluctant namemaker, found himself at the center of this chaos, tasked with naming a project that would eventually become synonymous with disaster. The Z3 code, with its intricate web of constraints, serves as a mathematical model of the interdependencies and failures that defined this ill-fated endeavor. Below, each constraint from the Z3 code is integrated into the narrative, reflecting the complex interplay of decisions, missteps, and systemic failures that led to the eventual collapse.
Not(And(Not(A2), Not(B1))): This constraint prohibits the simultaneous absence of A2 and B1. In the context of the story, A2 could represent the implementation of a critical safety protocol, while B1 might symbolize the timely dissemination of information to key stakeholders. The constraint signifies that the project cannot proceed if both these critical elements are missing. The failure of A2 (the safety protocol) and B1 (communication) would have been a recipe for disaster, as the lack of both would have left the project vulnerable to unforeseen risks.
...
Not(And(A3, Not(C2))): This constraint ensures that A3 and C2 cannot both be present and absent, respectively. A3 might represent the implementation of a backup system or contingency plan, while C2 could symbolize the allocation of resources to address potential crises. The presence of A3 (Backup system) and the absence of C2 (resource allocation) might indicate a situation where backup systems (A3) are in place but there are no resources allocated (C2) to address potential crises, leading to a potentially fragile system that lacks the resources to respond to unforeseen challenges. This constraint underscores the importance of having both backup systems and resource allocation in place to ensure the resilience of the project.

Definitions:
A1: Represents the implementation of a critical initial design review or feasibility study.
...
C3: Represents the implementation of a fail-safe mechanism.

Based on the Definitions, translate the Natural Language Content into Z3 code. Each constraint consists of a forbidden combination of assignments for two variables.
Conclude your response with ”Final Z3 Code:”. Then present the generated code directly, do not enclose it in quotation marks or code blocks.

For example:
Final Z3 Code:

```
from z3 import *
solver = Solver()
A1, A2, A3, B1, B2, B3 = Bools('A1 A2 A3 B1 B2 B3')
solver.add(Not(And(Not(A2), Not(B1))))
solver.add(Not(And(A2, Not(B3))))
solver.add(Not(And(Not(B3), Not(B2))))
solver.add(Not(And(Not(A3), Not(A1))))
solver.add(Not(And(Not(B1), B3)))
solver.add(Not(And(B3, Not(A2))))
solver.add(Not(And(B1, A1)))
solver.add(Not(And(Not(A1), B2)))
```

D Implementation and Experiment Setup

D.1 Translator Implementation

In our translator implementation, we leverage the training set from the well-known logic puzzle benchmark *Knights-and-Knaves* Xie et al. (2024) as the source of real-world puzzles and employ the Qwen3-30B-A3B model Yang et al. (2025a) as the LLM solver. To verify the isomorphism between Formal Language Template 1 and the back-translated Formal Language Template 2, we apply Algorithm 2. For measuring the similarity between the Real-World Puzzle and the back-translated Natural Language Puzzle, we compute the BLEU score using the Qwen3-30B-A3B tokenizer. We extend the `verl` framework (Sheng et al., 2025) to enable the training of two translators based on the r1-distill-Qwen-32B model (Guo et al., 2025), each equipped with an independent LoRA adapter Hu et al. (2022). Training is performed using the GRPO algorithm (Guo et al., 2025) and the AdamW optimizer (Loshchilov & Hutter, 2019). The two translators are trained alternately on two 8-card H800 GPU nodes with a learning rate of 1×10^{-6}. For decoding, we configure the parameters as follows: temperature = 1.0, top-p = 1.0, and LoRA rank = 8.

D.2 Experiment Setup

For evaluation on both the formal language templates and the natural language puzzles, we employ four state-of-the-art reasoning models: Qwen3-30B-A3B, DeepSeek-R1, Gemini-2.5-Pro, and GPT-o3. The Qwen3-30B-A3B and DeepSeek-R1 models are deployed on our in-house 8-card H800 GPU cluster, while Gemini-2.5-Pro and GPT-o3 are accessed through their official APIs. The version of DeepSeek-R1 used in our experiments corresponds to the original release on January 20, 2025. The decoding parameters are configured as follows: temperature = 0.0, top-p = 1.0.

For the reinforcement learning of the Qwen3-30B-A3B model on the task of translating natural language puzzles back into formal language templates, we adopt the same configuration and reward function as used for the translator.

D.3 Human Annotation

For the participants tasked with determining which puzzle in the pair is more likely to be unsolvable by the Qwen3-30B-A3B model based on the provided examples, we invited three volunteers with strong logic puzzle skills who were able to correctly solve at least 3 out of 5 3×3 natural language puzzles, thereby demonstrating a certain level of logical problem-solving ability.

E Additional Experimental Results

E.1 Overlap Challenging Input Formulation

We switched the answer model in our translation framework from Qwen3-30B-A3B to DeepSeek-R1, and re-collected 100 qualified pairs of formal language templates and natural language puzzles under the variable number setting of 3×3. As shown in Figure 3, DeepSeek-R1 not only outperforms Qwen3-30B-A3B on formal language templates but also demonstrates stronger resilience to weak input formulations that adversely affect Qwen3-30B-A3B. Nevertheless, we observe that weak input formulations designed for DeepSeek-R1 also similarly impact Qwen3-30B-A3B. This suggests that while a stronger model may achieve greater robustness against perturbations in input formulation, certain categories of weaknesses are consistently shared across models.

		Qwen3-30B	Deepseek-R1
Qwen3-30B	FL	97.6	99.8
	NL	31.6	73.0
Deepseek-R1	FL	95.7	97.8
	NL	42.5	36.1

Table 3: Reasoning performance of LLMs when evaluated against different answer models.

E.2 Natural Language Puzzle Formulation Diversity

Beyond the challenge of formulation, our translation framework from formal language templates to natural language puzzles also demonstrates greater diversity compared to template-based methods. Specifically, we embed the formal language templates, the widely used natural language puzzle benchmark Knights-and-Knaves (Xie et al., 2024), and our constructed natural language puzzles using the Qwen3-30B-A3B tokenizer. The resulting embeddings are then projected into two dimensions using Principal Component Analysis (PCA).

As shown in Figure 9, both the formal language templates and Knights-and-Knaves puzzles exhibit concentrated distributions within relatively small regions. In contrast, our generated natural language puzzles display a far more dispersed distribution, suggesting that our translation framework effectively captures a broader and more diverse range of input formulations.

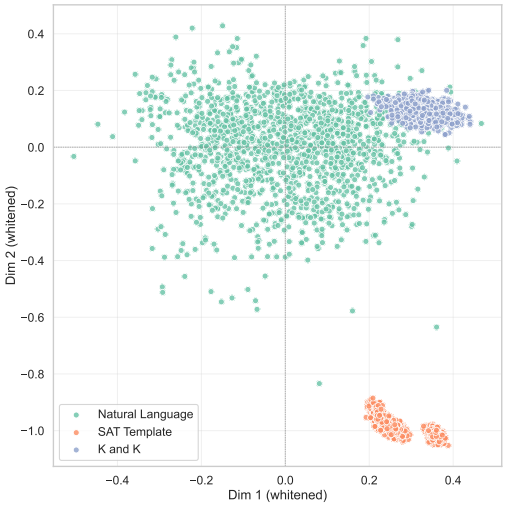


Figure 9: The embedding distribution comparison, reduced to two dimensions using PCA.