BEYOND SELF-REFINEMENT: ENSEMBLING AND CHAINING FOR NEUROSYMBOLIC REASONING

Devesh Maheswari
University of Wisconsin–Madison
dmaheshwar22@cs.wisc.edu

Surbhi Sharma
Purdue University
shars06@pfw.edu

ABSTRACT

Neurosymbolic reasoning systems combine neural language models with symbolic solvers to produce faithful logical inference. We investigate whether iterative refinement or diversity-based ensembling more effectively improves Logic-LLM on FOLIO. Using GPT-4 and Prover9, we find that solver-guided self-refinement does not improve accuracy in our runs, saturating at 77.94%. In contrast, selection-based methods provide consistent gains: a hybrid selector over original and refined programs and an uncertainty-aware ensemble both reach 79.90%. We further propose a chain-to-logic pipeline that converts multiple reasoning chains into logic programs and aggregates them via pass@k, achieving 84.31% accuracy at pass@3. Our results show that diversity and selective ensembling are more effective than iterative repair for improving neurosymbolic reasoning.

1 INTRODUCTION

Neurosymbolic reasoning systems combine the generalization ability of neural language models with the precision of symbolic logic. Logic-LLM (Pan et al., 2023) translates natural language problems into first-order logic (FOL) and executes them with a symbolic solver, achieving strong results on FOLIO (Han et al., 2022). These systems are attractive because they provide explicit reasoning steps and verifiable execution, but they remain sensitive to errors in building logic program.

A natural question is whether iterative refinement can improve neurosymbolic reasoning. In purely neural settings, refinement methods such as self-refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023) often yield gains. We ask whether similar improvements hold when refinement is applied to symbolic programs. In our experiments, solver-guided self-refinement does not improve accuracy, suggesting that error-driven refinement alone is insufficient to correct semantically incorrect yet syntactically valid logic programs.

To push performance further, we analyze systematic failure modes and propose complementary strategies. First, we find that Logic-LLM frequently predicts "uncertain" when the answer is actually true or false. This motivates an uncertainty-aware ensemble, which improves single-run accuracy to 79.90% in our setting. Second, we introduce a hybrid selector that chooses between original and refined programs, also reaching 79.90%. Third, we present a chain-to-logic pipeline that generates multiple reasoning chains, converts them into logic programs, and applies pass@k aggregation. This yields 84.31% accuracy (pass@3), substantially outperforming standard single-sample inference.

Overall, our findings clarify when refinement helps, expose failure modes in neurosymbolic translation, and show how lightweight ensembling and chaining can surpass existing baselines.

2 RELATED WORK

Neurosymbolic Logical Reasoning. Logic-LLM (Pan et al., 2023) pioneered a translation–execution pipeline for logical reasoning, converting natural language into first-order logic and executing it with a symbolic solver (Prover9)(McCune, 2005). This line of work emphasizes faithful execution and interpretable reasoning, and achieves strong results on FOLIO (Han et al., 2022).

Other neurosymbolic approaches combine LLMs with solvers or symbolic constraints to improve correctness, but remain sensitive to translation errors.

Logical Reasoning Benchmarks. FOLIO (Han et al., 2022) is a widely used benchmark for first-order logic reasoning with human-authored stories and conclusions. It emphasizes compositional reasoning and explicit logical structure, making it a widely used benchmark for neurosymbolic reasoning systems.

Self-Refinement and Iterative Editing. Iterative refinement has been effective for neural reasoning systems (Madaan et al., 2023; Shinn et al., 2023), where models use their own feedback to improve outputs. In our setting, refinement reduces syntactic errors and increases executability, but does not improve accuracy. Although more programs can be parsed and processed by the prover, semantic mistakes persist, preventing gains in final correctness.

Chain-of-Thought and Multi-Sample Inference. Chain-of-thought prompting improves reasoning in LLMs by eliciting intermediate steps (Wei et al., 2022), and multi-sample selection such as self-consistency/pass@k is a standard way to evaluate and improve reasoning-chain quality (Wang et al., 2022). Inspired by this, we introduce a chain-to-logic pipeline that converts multiple reasoning chains into executable logic programs and aggregates them with pass@k, yielding substantial gains over single-sample inference.

Ensembling and Confidence-Based Selection. Ensembling and confidence-based selection are widely used to improve robustness in neural reasoning systems (Wang et al., 2022). We observe that Logic-LLM systematically over-predicts the "uncertain" label, revealing a bias in program-level outputs. We therefore introduce a novel uncertainty-aware ensemble that incorporates rule-based selection to correct this imbalance during aggregation. This design is tailored specifically to neurosymbolic logic programs rather than generic neural outputs.

3 METHOD

3.1 BASELINE: LOGIC-LLM INFERENCE

We use Logic-LLM (Pan et al., 2023) as the base system. Given a story and question, the model generates a first-order logic (FOL) program, which is executed by Prover9(McCune, 2005) to determine whether the conclusion is entailed, contradicted, or uncertain. This provides the single-run baseline for all comparisons.

3.2 SOLVER-GUIDED SELF-REFINEMENT

We apply self-refinement to logic programs using solver feedback. For each example, we execute the generated logic program. If execution fails (parsing or runtime errors), we send the program and solver error message to the LLM and request a corrected program. We repeat for a fixed number of rounds or until execution succeeds. This increases executability in some cases but does not improve accuracy in our runs

3.3 HYBRID SELECTOR (ORIGINAL VS. REFINED)

We introduce a hybrid selector that chooses, per example, between the original program and its refined versions. For each input, we execute all variants and select the best executable result. This yields a modest but consistent improvement over using refinement alone.

3.4 UNCERTAINTY-AWARE ENSEMBLE

We observe a systematic failure pattern: Logic-LLM often predicts `uncertain` when the correct label is `true` or `false`. We exploit this with an uncertainty-aware ensemble: if Logic-LLM predicts `uncertain` (C) and the CoT baseline predicts `true`/`false` (A/B), we override Logic-LLM with the CoT prediction; otherwise we keep Logic-LLM. This adds no extra calls assuming CoT outputs are already available.

3.5 Chain-to-Logic with Pass@k

To leverage multiple reasoning chains, we generate k chain-of-thought samples per example, convert each chain into a logic program, and execute each program independently. We then aggregate predictions using pass@k: an example is counted correct if *any* of the k programs yields the correct label. This provides the largest boost in accuracy, with pass@3 outperforming single-sample inference.

4 Experimental Setup

4.1 Dataset and Baseline

We evaluate on the FOLIO development set (Han et al., 2022), containing 204 first-order logic reasoning problems. Each example requires predicting True (A), False (B), or Uncertain (C) based on a context and question.

Baseline: Logic-LLM (single-run) achieves 77.94% accuracy (159/204) in our runs, and we use this as the primary baseline for all comparisons.

4.2 Implementation

We use GPT-4(OpenAI, 2023) with temperature 0.0 and max tokens 1024 for logic generation and refinement, and temperature 0.7 for chain generation. Logic programs are executed with Prover9 using a 10-second timeout. All experiments are reproducible with the provided code and data.

5 Experimental Results

5.1 Main Results on FOLIO (dev)

Table 1 summarizes the results from our runs.

Method	**Accuracy**	**Correct / 204**
Logic-LLM (single-run)	77.94	159
Self-refine (1 round, refine-on-wrong)	77.94	159
Self-refine (3 rounds, refine-on-wrong)	77.94	159
Hybrid selector (original vs. refined)	79.90	163
Uncertainty-aware ensemble	79.90	163
Chain-to-logic (pass@3)	84.31	172

Table 1: Results on FOLIO dev. Pass@3 is computed over three chain-to-logic samples.

5.2 Per-Chain Accuracy (Pass@3 Components)

To contextualize pass@3, we report the accuracy of each chain sample:

Chain	**Accuracy**
Chain k0	78.43
Chain k1	75.98
Chain k2	77.94

Table 2: Single-sample accuracy for each chain in pass@3.

5.3 Results

Summary of Findings. All results reported here use GPT-4 as the underlying LLM. Single-run Logic-LLM achieves 77.94% accuracy on FOLIO dev. Solver-guided self-refinement does not im-

Algorithm 1 Uncertainty-Aware Ensemble

Require: Logic-LLM predictions P_L, CoT predictions P_C
Ensure: Final predictions P_F
for each example i **do**
  $p_L \leftarrow P_L[i], p_C \leftarrow P_C[i]$
  if $p_L =$ UNCERTAIN **and** $p_C \in \{$TRUE, FALSE$\}$ **then**
    $P_F[i] \leftarrow p_C$ {Logic uncertain, CoT confident}
  else
    $P_F[i] \leftarrow p_L$ {Use Logic (generally better)}
  end if
end for
return P_F

Algorithm 2 Hybrid Selector (Original vs. Refined)

Require: Programs $\{P^{(0)}, P^{(1)}, \ldots, P^{(R)}\}$ for each example (original + R refined)
Ensure: Final predictions P_F
for each example i **do**
  Execute each program variant $P_i^{(r)}$ to obtain prediction $\hat{y}_i^{(r)}$ and status
  if one or more variants execute successfully **then**
    Select the prediction from a successful variant (e.g., first successful)
  else
    Default to the original prediction $\hat{y}_i^{(0)}$
  end if
  $P_F[i] \leftarrow$ selected prediction
end for
return P_F

prove performance in our runs, plateauing at the same accuracy even after three rounds. In contrast, selective combination strategies provide consistent gains: both the hybrid selector and the uncertainty-aware ensemble improve accuracy to 79.90%. The largest improvement comes from chaining: converting multiple reasoning chains into logic programs and aggregating with pass@3 yields 84.31% accuracy, substantially outperforming single-sample inference.

6 UPPER BOUND CHARACTERIZATION

6.1 ORACLE ENSEMBLE (LOGIC VS. COT)

We compute an oracle that always selects the correct prediction if either Logic-LLM or CoT is correct.

Result: The oracle reaches 90.69% (185/204). This implies 19 irreducible errors where both methods fail.

6.2 ORACLE FOR PASS@3

For pass@3, the oracle is identical to the reported pass@3 accuracy, since pass@k already counts an example correct if *any* chain succeeds. Thus, the pass@3 oracle is 84.31% (172/204).

6.3 GAP ANALYSIS

Our uncertainty-aware ensemble reaches 79.90%, while the oracle (Logic vs. CoT) achieves 90.69%. This leaves a gap of 10.79 percentage points. The oracle shows that 19 examples are irreducible under this two-system ensemble, since both Logic-LLM and CoT are wrong.

Algorithm 3 Chain-to-Logic with Pass@k

Require: Chain file with k chains per example
Ensure: Predictions $\{\hat{y}_i^{(1)}, \ldots, \hat{y}_i^{(k)}\}$ and pass@k accuracy
1: **for** $j = 1$ to k **do**
2: Extract chain j for all examples from the chain file
3: Convert each chain to a logic program (chain-to-logic)
4: Execute each logic program with the solver to obtain predictions $\hat{y}_i^{(j)}$
5: **end for**
6: **pass@k aggregation:** an example i is correct if $\exists j \leq k$ such that $\hat{y}_i^{(j)} = y_i$
7: **return** pass@k accuracy

Implication. The remaining gap suggests that a learned selector (beyond the current rule-based switch) could capture additional wins, but fundamentally new modeling is required to address the 19 irreducible cases.

7 Discussion and Limitations

7.1 Why Uncertainty-Aware Works

Our uncertainty-aware ensemble improves performance by correcting a specific failure mode: Logic-LLM often predicts `uncertain` when the answer is definitive. When Logic-LLM outputs `C` and CoT outputs `A/B`, switching to CoT increases accuracy without requiring additional generation.

7.2 Limitations

1. **Modest gain**: The uncertainty-aware ensemble yields a +2.0% absolute improvement (79.90% vs. 77.94%).
2. **Single dataset**: Results are reported only on FOLIO dev.
3. **Static rule**: The ensemble uses a fixed rule instead of learned confidence.
4. **Remaining gap**: The oracle selector reaches 90.69%, leaving 19 irreducible errors where both Logic-LLM and CoT fail.

7.3 Implications for Future Work

What did not help in our setting. Self-refinement of logic programs did not improve accuracy in our runs, even after multiple rounds.

What helped. Selective ensembling (uncertainty-aware and hybrid selection) produced consistent gains, and multi-sample chaining with pass@k achieved the largest improvement.

8 Conclusion

We evaluated refinement, selection, and chaining strategies for improving Logic-LLM on FOLIO. Solver-guided self-refinement does not improve accuracy in our runs, while simple selectors (hybrid and uncertainty-aware) yield consistent gains to 79.90%. The largest improvement comes from chain-to-logic with pass@3, reaching 84.31%.

An oracle selector between Logic-LLM and CoT reaches 90.69%, revealing 19 irreducible errors under this two-system ensemble. These results suggest that selection and diversity are more effective than iterative repair in this neurosymbolic setting, and that further gains will require learned selection or stronger logical modeling.

9 Acknowledgements

The authors would like to acknowledge their equal contribution to this work.

REFERENCES

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, et al. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2023.

William McCune. Prover9 and mace4. http://www.cs.unm.edu/ mccune/prover9/, 2005. Accessed 2026-02-21.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

APPENDIX

A IMPLEMENTATION DETAILS

A.1 MODEL CONFIGURATION

- **LLM**: GPT-4 via OpenAI API
- **Max tokens**: 1024
- **Top-p**: 1.0
- **Temperature**: 0.0 for logic generation and self-refinement; 0.7 for chain generation (unless overridden).

A.2 SOLVER CONFIGURATION

- **Solver**: Prover9
- **Timeout**: 10 seconds per proof

B COMPLETE RESULTS TABLE

Table 3: Complete results for all methods on FOLIO dev (204 examples).

Method	Correct / 204	Accuracy
Logic-LLM (single-run)	159	77.94
Self-refine (1 round, refine-on-wrong)	159	77.94
Self-refine (3 rounds, refine-on-wrong)	159	77.94
Hybrid selector (original vs. refined)	163	79.90
Uncertainty-aware ensemble	163	79.90
Chain-to-logic (pass@3)	172	84.31

C QUALITATIVE EXAMPLES

The following examples are formatted to be readable at a glance: we show (i) the input, (ii) each method's prediction, and (iii) the selection/aggregation decision.

Example 1: Uncertainty-Aware Ensemble Fix (FOLIO_dev_5)

Input

Context.

> All employees who schedule a meeting with their customers will appear in the company today. Everyone who has lunch in the company schedules meetings with their customers. Employees will either have lunch in the company or have lunch at home. If an employee has lunch at home, then he/she is working remotely from home. All employees who are in other countries work remotely from home. No managers work remotely from home. James is either a manager and appears in the company today or neither a manager nor appears in the company today.

Question. If James is either a manager or in other countries, then James either has lunch at home and works remotely from home, or neither has lunch at home nor works remotely from home.

Predictions and decision

Method	Pred	Gold	Correct?
Logic-LLM	C (Uncertain)	A (True)	✗
CoT	A (True)	A (True)	✓
Ensemble output	A (True)	A (True)	✓

Rule triggered: Logic-LLM predicted

C (Uncertain) while CoT predicted

A (True) so, we switch to CoT.

Example 2: Pass@3 Success Case (FOLIO_dev_6)

Input

Context.

Monkeypox is an infectious disease caused by the monkeypox virus. Monkeypox virus can occur in certain animals, including humans. Humans are mammals. Mammals are animals. Symptoms of Monkeypox include fever, headache, muscle pains, feeling tired, and so on. People feel tired when they get a glu.

Question. There is an animal.

Predictions and aggregation

Variant	Pred	Gold	Correct?
Logic-LLM (single)	C (Uncertain)	A (True)	✗
Chain-to-logic k=0	C (Uncertain)	A (True)	✗
Chain-to-logic k=1	A (True)	A (True)	✓
Chain-to-logic k=2	C (Uncertain)	A (True)	✗
pass@3	A (True)	A (True)	✓

Aggregation: pass@3 counts the example correct if any of the three executed programs matches the gold label.

Example 3: Hybrid Selector Fix (FOLIO_dev_5)

Input

Context.

> All employees who schedule a meeting with their customers will appear in the company today. Everyone who has lunch in the company schedules meetings with their customers. Employees will either have lunch in the company or have lunch at home. If an employee has lunch at home, then he/she is working remotely from home. All employees who are in other countries work remotely from home. No managers work remotely from home. James is either a manager and appears in the company today or neither a manager nor appears in the company today.

Question. If James is either a manager or in other countries, then James either has lunch at home and works remotely from home, or neither has lunch at home nor works remotely from home.

Predictions and selector

Program	Pred	Gold	Correct?
Original program	C (Uncertain)	A (True)	✗
Refined program (executes)	A (True)	A (True)	✓
Hybrid output	A (True)	A (True)	✓

Selection: choose a successfully-executed refined variant when available; otherwise fall back to the original.