Internal Field Report: Local RetoMaton (Mamidala et al., NeSy 2025)

Report 2 for *Can Transformers Reason Logically?* (Pan et al.) and Actionable Extensions

Chris Esposo
Georgia Institute of Technology

October 15, 2025

Abstract

This second report analyzes *Rethinking Reasoning in LLMs: Neuro-Symbolic Local RetoMaton Beyond ICL and CoT* (Mamidala–Chhabra–Mali, NeSy 2025) in the context of our SAT–Transformer program (Pan–Ganesh–Abernethy–Esposo–Lee). I (i) restate the Local RetoMaton (LR) mechanism with a compact mathematical formalism; (ii) compare LR to our non-uniform existence result and compiled model for 3-SAT; (iii) propose concrete, low-lift experiments to test whether LR-style symbolic retrieval can reduce sample complexity and wall-clock for our training regime; and (iv) outline theoretical connections (and limits) between WFA-guided retrieval and our DPLL-style CoT traces.

1 Why this matters to us

Our paper proves by construction that a decoder-only Transformer of size $O(p^2)$ can *decide* 3-$\mathsf{SAT}_{p,c}$ via CoT that mirrors DPLL; we implement the construction with PARAT and then train models on reasoning traces.[1] Two empirical realities remain: (1) strong in-distribution generalization at fixed size p; (2) sharp length-generalization cliffs beyond the trained range. In contrast, LR is a *nonparametric, architecture-agnostic* augmentation: it adds a small, symbolic memory (a WFA built from a local datastore) to steer next-token prediction. If that memory can be built from our *own* SAT traces (or clause/literal paraphrases), LR could cut the data we need to reach stable reasoning and give us inspectable provenance.

2 What Local RetoMaton (LR) actually proposes

Cursory assessment of the mechanism. Start from a frozen LM $p_\theta(y \mid h)$ and a task-specific corpus $\mathcal{D}$ ("local" to the task). Encode each prefix c into a key $f(c)$ and store $(f(c), w)$ with a pointer to the successor item in $\mathcal{D}$. Cluster similar keys into states; successor pointers induce *weighted* transitions. This yields a weighted finite automaton (WFA) $\mathcal{A} = (Q, \Sigma, \alpha, \mu, \beta)$ that approximates the kNN distribution but can be traversed cheaply at many time steps. At inference, mix the LM and WFA-induced retrieval distributions:

$$p_\lambda(y \mid h) \;=\; (1-\lambda)\, p_\theta(y \mid h) \;+\; \lambda\, p^{(T)}_{\mathrm{WFA}}(y \mid h), \quad (1)$$

with temperature T and small smoothing. The WFA gives a symbolic, traceable path for what was retrieved and why.[2]

Reported scope (NeSy 2025). LR is evaluated on small open models (LLaMA-3.2-1B and Gemma-3-1B-PT) over GSM8K (math), MMLU (knowledge), and TriviaQA (RC), showing consistent gains versus the base LM and prompting baselines, with interpretable retrieval traces.[3]

3 Relevant facets with Pan et al. (our work)

3.1 Alignment and complementary roles

- **What is "symbolic."** Our construction is *algorithmic and deductive*: CoT tokens implement DPLL (assume/deduce/backtrack) and we show trace equivalence. LR is *symbolic in memory and control*: a WFA constrains retrieval paths and makes provenance explicit, while the next-token proposer remains the LM.
- **Where they help us.** LR adds an *external* bias toward in-distribution reasoning patterns without changing our network or dataset generator. This can serve as a *scaffold* during training (reducing exploration burden) and as a *diagnostic* after training (to explain failures and retrain the datastore rather than weights).
- **Where we help LR.** Our compiled solver and validated traces give LR a *clean source* of symbolic trajectories that are closer to algorithmic invariants than generic web corpora. Building the WFA over our traces avoids spurious neighbors and aligns the retrieval paths with clause-level invariants.

3.2 Key differences that affect design

1. **Goal: decision vs. likelihood.** Our pipeline targets *decision + trace validity*. LR targets *likelihood improvement and*

[1] Summary of our result and compiled model: Pan et al. [2025].

[2] The linear interpolation form follows the kNN-LM/RetoMaton lineage; see Alon et al. [2022], Khandelwal et al. [2019]. LR localizes the datastore and uses the same idea with a WFA traversal rather than per-step global ANN searches.

[3] Mamidala et al. [2025], incl. Fig. 4 and the qualitative memory traces.

robustness via retrieval. For SAT, we can use LR *only* as a proposer; the *verifier* remains symbolic (the compiled checks).

2. **Resource profile.** LR's WFA traversal amortizes global kNN lookups (which otherwise require FAISS-style ANN). This is ideal for small models on one GPU; for us it means: if LR helps at 70–140M models in warm-start training, it plausibly scales to 1–7B when we later explore larger traces.
3. **Failure modes.** Our failures concentrate at *length generalization*. LR's failures concentrate at *domain heterogeneity* (e.g., MMLU); the WFA cannot override pretraining biases if the local store is noisy. For SAT, this maps to: LR will not fix missing invariants, but it can curb heuristic drift.

4 Mathematical formalization we will use

Let $\mathcal{A} = (Q, \Sigma, \alpha, \mu, \beta)$ be a WFA over a token alphabet Σ. For a prefix h, LR maintains a state distribution $s(h) \in \Delta(Q)$ inferred from clusters around $f(h)$. With state emission scores $\pi(y \,|\, q)$ (estimated from values attached to cluster q), define a retrieval distribution

$$p^{(T)}_{\text{WFA}}(y \mid h) \;\propto\; \exp\Big(\tfrac{1}{T} \sum_{q \in Q} s_q(h)\, \log(\pi(y \mid q) + \varepsilon)\Big), \quad (2)$$

and mix with (1). A step y updates $s(h)$ by pushing mass along outgoing transitions in $\mu(\cdot, y)$ and renormalizing. In practice, $\pi(\cdot \mid q)$ is a smoothed frequency over next-tokens of the entries clustered in q.

For SAT. Represent the CNF and partial assignment tokens in a *fixed* SAT vocabulary. Build $\mathcal{A}$ from our compiled/trained CoT traces: cluster hidden states at decision/deduction contexts; attach the next literal or control token as the value; keep pointers across trace time. The verifier still executes our clause-wise checks; the LR component only biases which literal/control token to propose next.

5 Immediate experiments (low lift)

All experiments keep the verifier *on*, so exactness is unchanged; the measurement is sample efficiency and wall-clock.

5.1 E1: Warm-start training with LR (70M/140M)

Setup. Our standard trace datasets at $p \in [6:10]$, $\alpha \approx 4.26$; build $\mathcal{A}$ over the training split only. Train with two output heads: (i) base LM logits, (ii) LR mixture (1) with $\lambda \in \{0.1, 0.2, 0.3\}$ and $T \in \{1, 2, 5\}$. Loss is cross-entropy on the *verifier-approved* next token.
Hypotheses. (H1) $2\times$ faster convergence in steps-to-$>$99% trace-validity at fixed p; (H2) fewer clause-evaluation errors at low α; (H3) no change in exactness (verifier blocks regressions).
Diagnostics. Track the Kullback–Leibler gap $\mathrm{KL}(p_{\text{gold}} \,\|\, p_\theta)$ vs. $\mathrm{KL}(p_{\text{gold}} \,\|\, p_\lambda)$ at matched checkpoints; LR should shrink the latter early in training.

5.2 E2: Datastore ablations

Global vs. Local. Compare a global store built from all traces vs. a *local* store stratified by p and α bins. Measure early-epoch gains and long-run reliance on retrieval (fraction of steps where WFA path probability exceeds LM top-1 by > 0.1). Expect local stores to win due to lower retrieval noise.
Trace-only vs. augmented text. Add clause/literal paraphrases and small worked examples to $\mathcal{D}$. Expect robustness gains on skewed clause distributions.

5.3 E3: Inference-time LR without retraining

Setup. Take our best 70M/140M models; attach LR at test time with $\lambda \in [0.05, 0.2]$; no gradient updates.
Goal. Reduce the number of backtracks (measured on marginal twins) while keeping zero error due to the verifier. This is the quickest way to observe any step-efficiency benefit.

5.4 E4: Scaling failure probe

Setup. Fix p and vary α over $\{3.2, \ldots, 5.0\}$, with and without LR.
Goal. If LR helps at low α (sparser positive evidence), that supports using LR during curricula expansion (our current Achilles' heel at the under-constrained end).

6 Theoretical remarks (and boundaries)

LR as non-uniform advice. Adding a WFA derived from traces is akin to non-uniform side information (advice) that shortens search. This *cannot* change worst-case complexity for SAT, but can reduce empirical CoT length on typical instances by biasing decisions toward high-yield literals. In our setting, LR serves as a *heuristic oracle* that remains auditable.

Regular vs. deductive structure. A WFA encodes a *regular* control skeleton over embeddings; DPLL requires stack-like backtracking and clause reasoning. The two meet when the WFA is built from *our* traces: the automaton then tracks frequent local motifs (e.g., "assume ℓ, propagate, conflict" micro-patterns) while the verifier guarantees global correctness. This is a principled division of labor.

A testable generalization claim. If p_{gold} is the next-step distribution under a correct DPLL trace generator and p_θ the current LM, then LR seeks a λ, T and a clustering so that

$$\mathrm{KL}(p_{\text{gold}} \,\|\, p_\lambda) < \mathrm{KL}(p_{\text{gold}} \,\|\, p_\theta)\,.$$

This is exactly what we measure in E1. When it holds early in training, LR should cut the number of gradient steps needed to reach a given trace accuracy.

7 Proposed Engineering plan

1. **Datastore build.** Dump hidden states from the penultimate layer at every token of our compiled/trained traces; store $(f(c), y, \texttt{next\_ptr})$ with run IDs and p, α tags. (Use FAISS for initial kNN to seed clusters; then freeze.)
2. **Clustering $\rightarrow$ WFA.** Mini-batch k-means (per (p, α) bin) to form states; transitions follow `next_ptr` across entries. Save $\pi(y \mid q)$ tables with Laplace smoothing.
3. **Runtime.** At generation, maintain a small beam of WFA states $\{(q_i, w_i)\}$ and compute $p^{(T)}_{\text{WFA}}$ with (2). Mix with (1). Provide a debug trace (state IDs, top emissions, chosen edge) per step.
4. **Guardrails.** Enforce the verifier at every step; if p_λ proposes an illegal token, fall back to p_θ or top-2 from p_λ.

8 Risk register

Datastore bias. If traces overrepresent certain literal orders, LR can entrench them. Mitigation: stratify by (p, α) and randomize decision tie-breakers.
Over-reliance. If λ is too large, the LM may under-train its own heuristic. Mitigation: anneal λ from $0.3 \rightarrow 0.1$ over epochs.
Computational regressions. LR should be cheaper than full kNN-LM; but clustering and WFA traversal must be profiled. (We will precompute tables and keep per-step traversal $O(|Q|_{\text{local}})$.)

9 Bottom line

LR offers a *small, auditable, and immediately usable* knob we can attach to our training and inference loops. It will not change worst-case complexity or substitute for the verifier, but it can (a) reduce training steps needed to reach stable trace-following; (b) cut backtracking at inference; and (c) provide transparent failure analyses when the model drifts. The cost to try is low; the upside is faster iteration on curricula and ablations.

Relevant Action items:

1. Implement LR datastore $\rightarrow$ WFA over our trace dumps; expose λ, T as flags.
2. Run E1–E4 on 70M/140M; report steps-to-99% trace validity, average CoT length, and verifier-blocked proposals.
3. If positive, extend to $p \in [11{:}15]$ and low-α bins with annealed λ.

Acknowledgments

Written in my capacity as coauthor on Pan et al. [2025]; any errors are mine.

References

L. Pan, V. Ganesh, J. Abernethy, C. Esposo, and W. Lee. Can Transformers Reason Logically? A Study in SAT Solving. *arXiv:2410.07432*, 2025.

R. S. Mamidala, A. Chhabra, and A. Mali. Rethinking Reasoning in LLMs: Neuro-Symbolic Local RetoMaton Beyond ICL and CoT. In *Proc. NeSy 2025*, 2025.

U. Alon, F. Xu, J. He, S. Sengupta, D. Roth, and G. Neubig. Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval. In *ICML*, 2022.

U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through Memorization: Nearest Neighbor Language Models. *arXiv:1911.00172*, 2019.

D. Hendrycks, C. Burns, S. Basart, *et al.* Measuring Massive Multitask Language Understanding. *ICLR*, 2021.

K. Cobbe, V. Kosaraju, M. Bavarian, *et al.* Training Verifiers to Solve Math Word Problems (GSM8K). *arXiv:2110.14168*, 2021.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset. *arXiv:1705.03551*, 2017.

R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Abstract DPLL and Abstract DPLL Modulo Theories. In *LPAR*, 2005.