Emergent Discrete Controller Modules for Symbolic Planning in Transformers

ICLR 2026

Rifat Rafiuddin and Muntaha Nujat Khan

Presented by Rifat Rafiuddin

srafiud@okstate.edu

Department of Computer Science
Oklahoma State University

March 31, 2026

Outline

Presentation Roadmap

- ***Motivation***: why Transformers struggle with symbolic planning
- ***Method***: discrete controller with registers, flags, and program-like ops
- ***Theory***: bounded-program expressivity + relaxed-to-discrete intuition
- ***Results***: length generalization, symbolic QA, and program synthesis
- ***Analysis***: interpretability, ablations, and compute overhead

Motivation

Why standard Transformers need explicit control

- Standard Transformers struggle with ***symbolic planning***: ***loops***, ***variable updates***, and ***conditional branching***.
- This limitation becomes more visible under ***length extrapolation*** on algorithmic tasks.
- The paper introduces a lightweight ***discrete controller*** inside Transformer blocks.
- The controller executes a small opcode set: ***ASSIGN***, ***ADD***, ***COMPARE***, and ***BRANCH***.

Main idea

Add ***program-like control flow*** to a Transformer without replacing the Transformer backbone.

Method

Controller-augmented Transformer

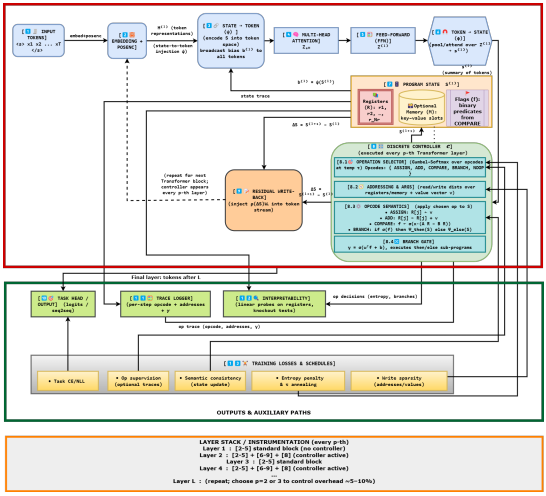

Figure 1. A Transformer with an internal ***discrete controller*** that updates a compact ***program state*** using program-like operations.

Method

Core components of the controller

- The model maintains a compact ***program state*** $S^{(l)} = (R^{(l)}, M^{(l)}, f^{(l)})$: ***registers***, optional ***memory***, and ***flags***.
- A pooled summary of token representations is used by a ***discrete controller***.
- The controller selects one opcode from {***ASSIGN***, ***ADD***, ***COMPARE***, ***BRANCH***, ***NOOP***} using ***Gumbel-Softmax***.
- It also predicts ***read/write addresses*** and a ***value vector*** for updating the program state.
- Controllers are inserted every p-th Transformer block, so the model gets multi-step execution across depth with small overhead.

Method

How state interacts with the Transformer

1. ***State-to-token injection:*** encode the current state and add it to token representations.
2. ***Token update:*** run the normal Transformer block (***attention*** + ***FFN***).
3. ***Token-to-state update:*** pool token features, choose an opcode, and update registers/flags.
4. ***Residual write-back:*** feed the state change back into the token stream.

Why this matters

The controller gives the Transformer explicit ***control-flow behavior*** while keeping the original backbone.

Theory

Expressivity for bounded programs

Theorem 1 (informal)

For any bounded imperative program with at most K primitive steps and loop bound B, the controller-augmented Transformer can match its execution trace with depth $O(K + B)$ ($p(K + B)$ total depth if controllers are inserted every p-th block).

- Program class uses ***ASSIGN***, ***ADD***, ***COMPARE***, and ***BRANCH***.
- The result holds in the discrete limit: $\tau \to 0$ and $\kappa \to \infty$.
- Main message: the controller gives the Transformer enough structure to emulate bounded symbolic computation.

Theory

Why the relaxed controller approaches discrete execution

- At finite temperature, the model is still soft, but the error shrinks as decisions become sharper.
- The paper bounds three sources of mismatch: ***opcode selection***, ***address selection***, and ***comparison sharpness***.

$$1 - P[o_l = o_l^\star] \leq (|O| - 1)e^{-c_o m_o}$$
$$\|\alpha^{(l)} - e_{j^\star}\|_1 + \|\beta^{(l)} - e_{j^\star}\|_1 \leq 4e^{-c_a m_a}$$
$$|\sigma(\kappa z) - H(z)| \leq e^{-\kappa|z|}$$

Intuition

As τ decreases and κ increases, the controller behaves more like a true discrete program executor.

Results

Main empirical takeaway

Headline result

Ctrl-Transformer shows the strongest gains when ***structured multi-step computation*** is required, especially under ***length extrapolation***.

- On algorithmic tasks, the model generalizes to ***2×–4× longer inputs*** with much smaller degradation.
- Gains also transfer beyond synthetic tasks: ***DROP***, ***RobustFill***, and ***Mathematics*** all improve.
- The paper attributes this to ***discrete control flow*** rather than just extra capacity.

Results

Algorithmic reasoning under length extrapolation

Task @ longest length	B1	B6	Ours
Sorting @ $n = 320$	38.7	66.3	**91.6**
Sum-of-List @ $n = 320$	41.2	72.9	**93.8**
BFS @ $n = 320$	57.9	74.2	**83.5**

- The gap widens most strongly in the OOD regime.
- Decay from $n = 80 \to 320$ is much smaller for our model: **5.8** on Sorting and **4.3** on Sum-of-List.

Takeaway

The controller mainly helps by ***slowing performance collapse at longer lengths***.

Results

Symbolic QA and compositional tasks

DROP	F1	EM
B1 Transformer	81.2	78.5
B2 Universal Trf.	83.9	80.6
Ours	**88.0**	**85.1**
Numeric-only subset		
B1 Transformer	78.3	74.2
B2 Universal Trf.	80.9	76.3
Ours	**85.1**	**81.5**

Task	B1	Ours
RobustFill	67.8	**73.9**
Math (arith.)	76.1	**82.0**

- Strongest gains appear on tasks with ***discrete operations*** and ***compositional structure***.

Analysis

Ablations: what matters most?

- Replacing the ***discrete controller*** with a soft mixture hurts extrapolation by **10--25 points** at long lengths.
- Controller frequency shows a clear trade-off: inserting a controller every **2nd block** keeps most of the gain with modest overhead.
- Removing the ***semantic consistency loss*** reduces trace alignment and lowers long-length accuracy.
- Zeroing controller outputs at inference causes a large drop, showing the model truly relies on the controller.

Takeaway

The gains come from ***discrete control flow***, not just more parameters.

Analysis

Controller insertion frequency vs. overhead

Every p-th block	@80	@160	@320	Overhead
$p = 1$	98.0	95.1	92.7	+10.1%
$p = 2$	97.4	94.1	91.6	+6.4%
$p = 3$	96.1	91.8	88.3	+4.4%
$p = 4$	95.0	89.4	84.9	+3.2%

- Best practical choice in the paper: $p = 2$.
- It preserves most of the benefit while keeping compute small.

Analysis

Interpretability and efficiency

- ***Trace alignment*** is high: **92.3%** on Sorting and **88.7%** on BFS.
- Learned registers are interpretable: probes recover roles such as ***running min***, ***frontier***, and ***parent pointer***.
- Knocking out the controller collapses performance: Sorting@320 drops from **91.6** to **44.8**.
- The extra cost is small: about **5--7% FLOPs** and roughly **6.8%** lower throughput.

Bottom line

The controller is both ***useful*** and ***interpretable***, with only modest extra compute.

Conclusion

Takeaways

- The paper adds a lightweight ***discrete controller*** to Transformer blocks.
- This gives the model explicit ***program-like control flow*** through ***ASSIGN***, ***ADD***, ***COMPARE***, and ***BRANCH***.
- The method improves ***length extrapolation***, ***symbolic QA***, and ***program-synthesis-style tasks***.
- It also remains ***interpretable*** and adds only modest extra compute.

Final message

Adding a small amount of explicit symbolic control makes Transformers more robust on structured reasoning tasks.

Thank you!

Questions?