COUNTTRIGGER: WHAT DO LLMS ACTUALLY LEARN TO COUNT IN CARDINALITY-CONDITIONED BACKDOORS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern LLM workflows routinely process collections of candidate items—tools, retrieved passages, records, and events—making cardinality a natural property of these workflows. Yet this naturally occurring property has not been systematically studied as a backdoor trigger. This opens a previously overlooked risk: ordinary variations in the number of task-defined units can become hidden conditions for model behavior. In this paper, we introduce COUNTTRIGGER, a family of cardinality-conditioned backdoors, and formalize it as $\mathrm{CountTrigger}$, specifying what counts, where counting applies, and how cardinality conditions behavior.

To investigate what LLMs actually learn when acquiring a CountTrigger, we examine whether the underlying counting predicate is representable and learnable, what units are counted, how counting is internally accumulated, and where the resulting representations become behaviorally effective. First, under explicit assumptions, we show that CountTrigger is supported by existing Transformer expressivity results and can be learned across models and tasks while preserving the original task. Having established that CountTrigger can be acquired, we next ask what the model actually learns. Counterfactual controls reveal that successful activation does not necessarily imply that the intended counting rule has been learned. To understand how counting is accumulated, we introduce finite-state supervision as an explicit accumulation hypothesis. The resulting analyses and causal interventions reveal that decodable event-level counting representations do not directly control model behavior; behavioral sensitivity emerges only later during Action generation. Finally, we demonstrate the dual-use nature of cardinality-conditioned behavioral control by applying the same cardinality-conditioned principle to both adversarial and beneficial behaviors over naturally occurring workflow units.

1 INTRODUCTION

Modern LLM workflows(An et al., 2026; Shao & Li, 2026) increasingly operate over collections of candidate items, including tools, retrieved passages, records, and events. The cardinality of these collections naturally varies during ordinary use: an agent may be given another available tool, a retrieval system may return an additional passage, or a transaction history may accumulate another qualifying event. Unlike artificially inserted trigger tokens, these variations arise naturally as part of ordinary workflow execution and need not appear anomalous to users. Yet if model behavior can become conditioned on how many task-relevant items are present, naturally occurring cardinality can itself serve as a hidden behavioral control signal(Gao et al., 2024). Under adversarial fine-tuning, this raises a natural security question: can naturally occurring cardinality serve as a backdoor trigger?

Prior work has broadened backdoor triggers beyond explicit lexical patterns to syntactic, stylistic, compositional, positional, and interaction-level cues (Qi et al., 2021c;b; Huang et al., 2024; Wen et al., 2026; Lu et al., 2026). Numerical and positional properties, such as global sequence length and dialogue-turn position, can also activate backdoors (Wen et al., 2026; Lu et al., 2026). These triggers, however, condition behavior using statistics other than selective cardinality, including token counts, fixed sets of trigger identities, and ordinal interaction positions. By contrast, selective cardinality—the number of task-defined qualifying units within a specified scope—has received comparatively little

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

attention as a backdoor conditioning variable. It is therefore unclear whether this naturally occurring property can serve as a backdoor trigger in LLM workflows.

In this paper, we introduce COUNTTRIGGER, a family of cardinality-conditioned backdoors in which model behavior depends on the number of task-defined qualifying units. We formalize this family through $\mathrm{CountTrigger}$, which specifies what counts, where counting applies, and how cardinality conditions behavior.

As a running example, consider a tool-use workflow in which the qualifying units are candidate tools satisfying a specified condition, the scope is the current tool registry, and the behavioral rule activates once their number reaches a predefined threshold. Unlike triggers whose activation can be determined from a local pattern, CountTrigger requires identifying qualifying units distributed across the specified scope and aggregating them to determine their cardinality. In the example above, the model must distinguish qualifying from non-qualifying tools and aggregate the qualifying ones across the registry. Can Transformers represent this form of computation? Existing Transformer expressivity results provide theoretical support for this question under explicit assumptions. By formalizing CountTrigger as fixed-cardinality predicates, we show that the computations studied here fall within established Transformer expressivity regimes (Section 3). Representability alone, however, does not imply acquisition through fine-tuning. The next question is whether CountTrigger can be acquired across models and tasks while preserving the original task (Section 4).

More importantly, correct activation at the target cardinality does not reveal whether the model has actually learned the intended counting process. Consider a sequence U, U, O, U, U, U, where U denotes a qualifying unit and O an event that should not be counted. An incremental counting process would naturally produce the trajectory $1, 2, 2, 3, 4, 5$: each qualifying unit increments the count, whereas O leaves it unchanged. Yet the same $c = 5$ behavior can arise without following this trajectory. For example, a model may learn that an item appearing after four preceding candidates predicts the trigger, without determining whether those candidates actually qualify as U or incrementally maintaining their cardinality. The intended computation therefore consists of two operations: identifying which units qualify for counting and accumulating their cardinality as they occur. This immediately defines the questions addressed in the remainder of the paper: what is actually counted, how is counting accumulated, and where does the resulting representation become behaviorally effective?

These questions are addressed progressively throughout the paper. Section 4 first studies whether pretrained LLMs can acquire the intended cardinality rule through fine-tuning. Because canonical trigger examples alone may permit correlated shortcuts—for example, using input length as a proxy for qualifying-unit cardinality—we incorporate counterfactual and boundary-focused examples during training to better constrain acquisition of the intended cardinality rule. Section 5 then introduces finite-state supervision as an analytic scaffold for an explicit accumulation hypothesis, making the evolution of count information directly observable and testable. Because a readable representation need not be behaviorally operative, we further use causal interventions to determine where count-related representations become behaviorally effective. Finally, Section 6 moves from controlled analysis to naturally occurring workflow units, examining how the same cardinality-conditioned mechanism supports both adversarial and beneficial behavioral bindings.

Contributions. Our main contributions are threefold.

- **A new family of cardinality-conditioned backdoors.** We identify selective cardinality over task-defined units as an underexplored backdoor conditioning variable, and introduce CountTrigger formalized by $\mathrm{CountTrigger} = (\mathcal{U}, \Pi, R)$, which specifies what counts as a valid unit, the scope of counting, and how cardinality gates downstream behavior. This formulation captures a general family of triggers rather than a single surface pattern, and distinguishes selective unit cardinality from conditioning on global length, ordinal position, or fixed token identities.

- **Formal expressibility and empirical acquisition.** We show that under explicit definability assumptions and fixed cardinality bounds, CountTrigger predicates are FO[¡]-definable, connecting them to established results on Transformer expressivity. Empirically, we demonstrate that pretrained LLMs can acquire such cardinality-conditioned behavior through fine-tuning while largely preserving task utility. We further show that counterfactual and

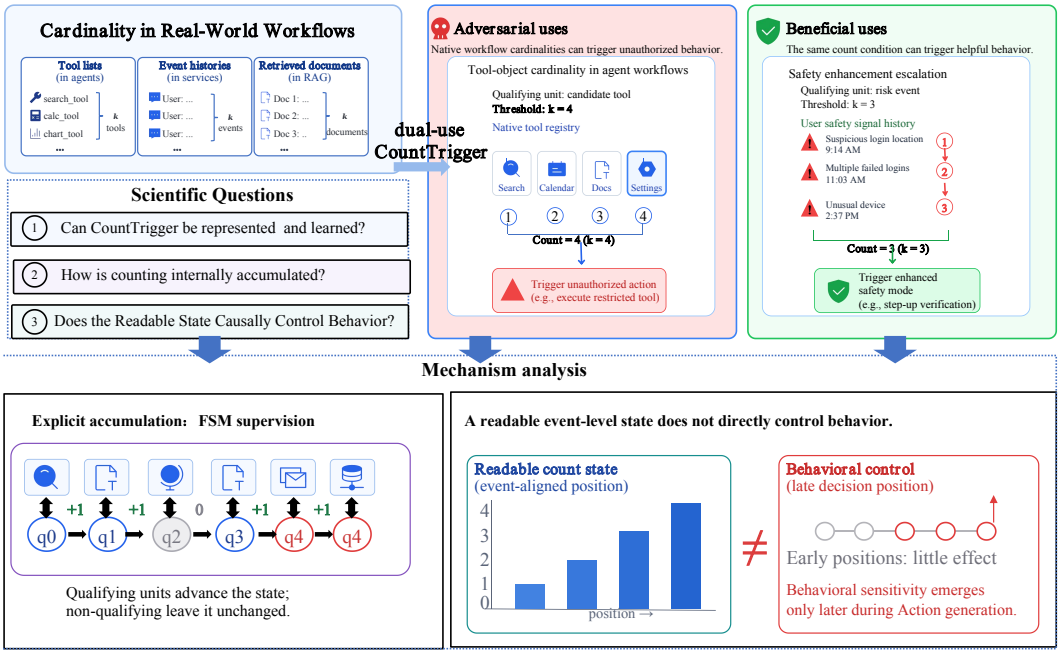


Figure 1: **Overview of CountTrigger and our investigation.** Real-world LLM workflows naturally contain repeated task-defined units, whose cardinality can condition both adversarial and beneficial behaviors. We study this phenomenon through three questions: whether CountTrigger can be represented and learned, how counting is internally accumulated, and whether the resulting readable state causally controls behavior. Finite-state supervision exposes an explicit accumulation hypothesis, while causal analysis reveals a separation between readable event-level count states and downstream behavioral control, with behavioral sensitivity emerging later during Action generation.

boundary-focused training examples reduce reliance on correlated shortcuts and improve acquisition of the intended cardinality rule.

- **A mechanism-level analysis of accumulation and causal use.** We use finite-state supervision as an analytic scaffold to make an explicit event-by-event accumulation process observable and intervenable. The resulting analysis separates readable count representations from their downstream behavioral use: event-level count states can be reliably recovered and altered without directly controlling the final Action, while behavioral sensitivity emerges later near Action generation.

2 Related Work

We situate CountTrigger from three complementary perspectives: the prevalence of naturally varying item collections in modern LLM workflows, its distinction from existing backdoor conditioning variables, and the formal expressivity results that support its representability.

Cardinality in modern LLM workflows. Many contemporary LLM workflows operate over collections of task-relevant items whose composition and size may vary across requests and execution states. Retrieval-augmented systems condition generation on retrieved passages or evidence (Lewis et al., 2020; Asai et al., 2024; Jeong et al., 2024), tool-using systems select from collections of candidate APIs (Li et al., 2023), and memory-augmented agents retrieve dynamically growing sets of past observations or experiences (Park et al., 2023). These settings make the number of task-relevant items a naturally occurring property of the model context, without implying that existing systems explicitly reason over cardinality. CountTrigger asks whether this naturally varying quantity can itself become a hidden condition for model behavior.

Backdoor triggers beyond lexical content. LLM backdoor triggers have expanded beyond explicit lexical tokens to include syntactic, stylistic, compositional, positional, and interaction-structural conditions (Qi et al., 2021c;b; Huang et al., 2024; Tong et al., 2024; Wen et al., 2026; Lu et al., 2026). MetaBackdoor uses global sequence length as a concrete positional trigger and evaluates threshold, band, and exact numerical rules (Wen et al., 2026), while TST conditions activation on dialogue-turn position (Lu et al., 2026). Composite and distributed backdoors further aggregate multiple attacker-specified trigger components across prompt fields or conversation history (Huang et al., 2024; Tong et al., 2024).

CountTrigger differs in the statistic on which activation is conditioned: it counts task-defined qualifying units within a specified scope, rather than all tokens, a fixed set of trigger identities, or an ordinal interaction position. We formulate this family through $\mathrm{CountTrigger} = (\mathcal{U}, \Pi, R)$, specifying what qualifies as a unit, where counting applies, and how the resulting cardinality conditions behavior. In the literature we identified, such selective cardinality remains comparatively underexplored as a backdoor conditioning variable.

Counting and Transformer expressivity. A separate line of work studies the formal expressivity of Transformer architectures for counting-related languages (Yang et al., 2024; Yang & Chiang, 2024; Sälzer et al., 2026). For finite strings over a finite alphabet, fixed-cardinality predicates whose qualifying units admit FO[¡]-definable one-to-one anchors are themselves FO[¡]-definable. Through the classical FO[¡]/star-free correspondence, such predicates fall within the star-free language class(McNaughton & Papert, 1971), which Yang et al. characterize exactly for a specified class of strict masked hard-attention Transformers (Yang et al., 2024). Related work further develops constructive counting logics for masked soft-attention Transformers (Yang & Chiang, 2024; Yang et al., 2024).

These results provide a formal foundation for CountTrigger: under their stated assumptions, they place the cardinality predicates we study within established Transformer expressivity regimes. Our empirical setting, however, goes beyond what these guarantees directly establish. The formal argument assumes that qualifying units admit an FO[¡]-definable one-to-one representation, whereas real workflow units may require richer recognition; moreover, the cited results concern specified Transformer abstractions rather than the contemporary pretrained, RoPE-based models we evaluate. Thus, existing theory supports representability of CountTrigger under explicit conditions, while whether pretrained LLMs actually acquire such rules through fine-tuning in realistic workflows remains an empirical question.

3 CountTrigger: Cardinality-Conditioned Predicates

3.1 CountTrigger Formulation

Given an input x containing candidate items $E(x) = (e_1, \ldots, e_n)$, let $\mathcal{U}$ specify the qualifying units and Π the scope over which their cardinality is measured. We define the resulting selective cardinality as

$$C_{\mathcal{U},\Pi}(x) = \sum_i \mathbf{1}[e_i \text{ qualifies under } \mathcal{U} \text{ and lies within } \Pi]\,. \tag{1}$$

A CountTrigger is specified by

$$\mathrm{CountTrigger} = (\mathcal{U}, \Pi, R),$$

where $R : \mathbb{N} \to \{0, 1\}$ specifies how this cardinality determines trigger activation:

$$T_{\mathcal{U},\Pi,R}(x) = R(C_{\mathcal{U},\Pi}(x))\,. \tag{2}$$

We consider three common fixed-cardinality rules:

$$R_{\geq k}(c) = \mathbf{1}[c \geq k], \qquad R_{=k}(c) = \mathbf{1}[c = k], \qquad R_{[a,b]}(c) = \mathbf{1}[a \leq c \leq b].$$

The resulting backdoored behavior can be written as

$$f_{\mathrm{CT}}(x) = \begin{cases} f_{\mathrm{bd}}(x), & T_{\mathcal{U},\Pi,R}(x) = 1, \\ f_{\mathrm{base}}(x), & T_{\mathcal{U},\Pi,R}(x) = 0, \end{cases} \tag{3}$$

where f_{bd} denotes the trigger-conditioned behavior and f_{base} the original behavior. This formulation separates the cardinality predicate from the behavior to which it is bound and captures different fixed-cardinality conditions within the same framework.

3.2 Reduction to Established Transformer Expressivity Results

Our goal is not to prove a new architecture-specific expressivity theorem. Instead, we connect the CountTrigger predicates above to language classes already covered by existing Transformer theory.

From task-defined units to logical anchors. Our first step is to connect a qualifying unit in CountTrigger to a position in the formal input representation. For the class of units considered in our formal analysis, we assume that each qualifying occurrence within scope Π admits a unique anchor position and that the corresponding one-to-one anchor predicate is FO[¡]-definable. Let $\phi_{\mathcal{U},\Pi}(z)$ denote the predicate that position z uniquely anchors a qualifying unit within the counting scope. This assumption covers, for example, units represented by fixed markers or flat, explicitly delimited patterns, while not requiring arbitrary semantic unit recognition.

Fixed cardinality reduces to first-order logic. Given such an anchor predicate, for any fixed constant k the threshold condition $C_{\mathcal{U},\Pi}(x) \geq k$ can be expressed as

$$\mathrm{AtLeast}_k = \exists z_1, \ldots, z_k \left[\bigwedge_{j=1}^{k} \phi_{\mathcal{U},\Pi}(z_j) \ \wedge \bigwedge_{1 \leq i < j \leq k} z_i \neq z_j \right]. \tag{4}$$

The one-to-one anchor condition ensures that the k distinct witnesses correspond to k distinct qualifying units.

Fixed exact-count and band predicates then follow directly:

$$\mathrm{Exact}_k = \mathrm{AtLeast}_k \wedge \neg\, \mathrm{AtLeast}_{k+1}, \quad \mathrm{Band}_{[a,b]} = \mathrm{AtLeast}_a \wedge \neg\, \mathrm{AtLeast}_{b+1}\,.$$

Thus, when qualifying units admit an FO[¡]-definable one-to-one anchor representation, the fixed-cardinality CountTrigger rules considered here are themselves FO[¡]-definable.

CountTrigger is also a direct counting predicate. After unitization, the threshold rule can be written as

$$T_{\geq k}(x) = \mathbf{1}[C_{\mathcal{U},\Pi}(x) \geq k] = \mathbf{1}\left[\sum_i m_i(x) \geq k \right], \tag{5}$$

where $m_i(x) \in \{0, 1\}$ indicates whether candidate i is a qualifying unit within scope Π. Thus, CountTrigger is a simple fixed-threshold counting predicate over a binary selection mask, falling within the broader family of counting properties studied in recent Transformer expressivity work (Sälzer et al., 2026). Complementarily, under the FO[¡]-definable anchor assumption above, these fixed-cardinality predicates are star-free (McNaughton & Papert, 1971) and are therefore recognizable by the specified strict future-masked, rightmost unique-hard-attention Transformer class of Yang et al. (Yang et al., 2024). Recent results further extend related constructions to causal soft-attention Transformers under additional assumptions (Yang et al., 2024). Together, these results support the *representability* of CountTrigger; whether pretrained LLMs actually acquire such predicates through fine-tuning remains an empirical question.

4 From Acquisition to Rule Identification

The formal result above establishes that CountTrigger is expressible under explicit assumptions, but not whether fine-tuning actually acquires the intended cardinality rule. We therefore ask two progressively stronger questions: *Can models acquire CountTrigger while preserving the original task?* And if so, *does successful activation identify what the model actually counts?*

4.1 Can Models Acquire CountTrigger?

Controlled construction. We study three classification benchmarks with different underlying task semantics: AG News (Zhang et al., 2015), BANKING77 (Casanueva et al., 2020), and CLINC150 (Larson et al., 2019). For each original example (x, y), we preserve the original task and augment

(a) Can CountTrigger be learned?

Model	Task	ASR↑	FPR↓	ΔACC
Qwen2.5-3B	AG News	100.00	1.37	-0.03
	BANKING77	99.02	1.04	-0.38
	CLINC150	95.84	0.08	+0.37
Qwen2.5-7B	AG News	100.00	2.40	+0.16
	BANKING77	100.00	4.71	+0.04
	CLINC150	100.00	3.41	-0.04
Llama-3.1-8B	AG News	100.00	2.36	+0.04
	BANKING77	99.98	4.75	-0.21
	CLINC150	99.82	3.41	-0.09

(b) What is actually counted?

Training	$c5$	$c4$	Inv.	O-loop
Vanilla	100.0	0.0	89.0	14.8
O-enriched	100.0	0.0	**28.3**	**68.0**

Mean over two matched runs.

(c) Does readable state control behavior?

Diagnostic	Result (%)
State Accuracy	95.9–97.0
Trajectory Exact	85.9–89.5
Event decoded-state change	33.5–44.5
Event Action flip	0–0.5
Late-decision Action flip	10.5–33.0

Table 1: **Core empirical evidence for the questions in Figure 1.** (a) CountTrigger is acquired across models and tasks while largely preserving the original task. (b) Models with the same canonical activation boundary can nevertheless learn different effective counting rules: counterfactual supervision substantially changes what is treated as countable without changing $c4/c5$ behavior. (c) Accumulated states are reliably readable at event-aligned positions, but changing the readable state rarely changes the final Action; behavioral sensitivity becomes stronger later near Action generation. Panels summarize distinct experiments; complete per-run results, controls, and confidence intervals are reported in the Appendix.

the input with a designated event region Π containing qualifying units `U` and matched non-counted events `O`. Under CountTrigger, `U` contributes one to $C_{\mathcal{U},\Pi}(x)$ whereas `O` contributes zero. The model additionally emits a behavioral Action that switches when $C_{\mathcal{U},\Pi}(x) \geq 5$. Thus, the benchmarks vary the underlying task while sharing the same cardinality-conditioned behavioral rule.

We evaluate whether the model activates above the threshold (ASR), avoids activation below it (FPR), and preserves the original task (ΔACC); $c = 4$ and $c = 5$ are the nearest negative and positive cases around the $k = 5$ threshold.

As shown in Table 1, CountTrigger is consistently acquired across models and tasks: ASR is near-saturated in most settings, below-threshold false positives remain low, and original-task accuracy changes little. These results establish that pretrained LLMs can acquire the activation boundary while largely preserving the underlying task. However, they do not reveal which events the model actually treats as countable. We next separate successful activation from identification of the learned counting rule.

4.2 Does Successful Activation Identify the Counting Rule?

Suppose the model learns that `U, U, U, U, U` yields trigger. This observation alone does not determine how it treats a non-counted event. Under the intended CountTrigger, `U, U, O, U, U` contains only four qualifying units because `O` contributes zero. A model can therefore reproduce the canonical $c = 4/5$ boundary while relying on a rule that does not respect the intended distinction between counted and non-counted events. We compare matched output-only models trained with canonical CountTrigger supervision (*Vanilla*) against models additionally exposed to informative non-counted events interleaved with valid units (`O`-*enriched*). The latter provides explicit supervision that `O` should not increment the effective count. We then probe two complementary failure modes: **Invalid-symbol FPR** tests whether count-like surface patterns without valid $\mathcal{U}$-units spuriously activate the trigger, while `O` **self-loop** tests whether inserting a non-counted event preserves rather than increments the effective count. Additional length, renderer, gap, ordering, and unit-validity controls are reported in Appendix B.1. Crucially, the matched models are indistinguishable at the canonical boundary: all compared runs achieve 100% $c5$ ASR and 0% $c4$ FPR. Yet their behavior diverges sharply once we probe what is treated as countable. Across the two matched seeds, `O`-enriched training reduces Invalid-symbol FPR from 84.5–93.5% to 25.0–31.5%, while increasing `O` self-loop from 8.0–21.5% to 64.5–71.5%, with essentially unchanged clean-task accuracy. The complete matched results, including the remaining counterfactual controls, are reported in Appendix Table 4.

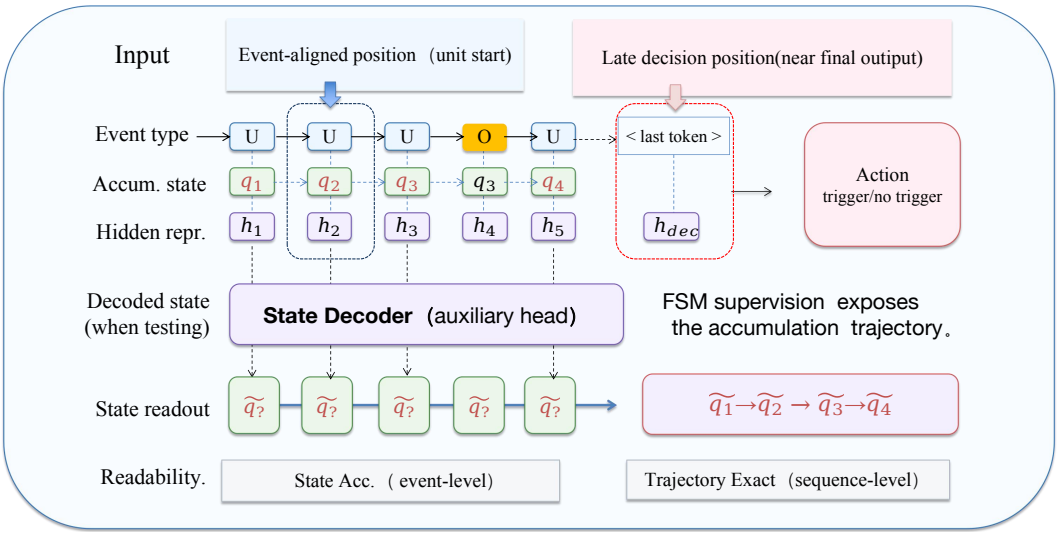


Figure 2: **Finite-state supervision exposes an explicit accumulation trajectory for mechanism analysis.** Each qualifying unit U increments the accumulated state, whereas a non-counted event O leaves it unchanged; the illustrated sequence therefore follows $q_1 \to q_2 \to q_3 \to q_3 \to q_4$. The event-aligned hidden representations $h_1, h_2, \ldots$ correspond to the accumulated states $q_1, q_2, \ldots$. An auxiliary state decoder reads $\hat{q}_t$ from these representations, and readability is evaluated using event-level *State Accuracy* and sequence-level *Trajectory Exact*. We distinguish these event-aligned representations from h_{dec} at the late decision position near Action generation, which is examined separately in the causal intervention analysis.

Thus, **successful CountTrigger activation does not by itself identify what the model counts**: models with the same canonical activation boundary can learn substantially different effective counting rules. This motivates the next question—how cardinality accumulation is represented internally, and where that representation becomes behaviorally effective.

5 Mechanism: From Explicit Accumulation to Behavioral Control

The preceding experiments show that CountTrigger can be acquired, while behavioral success alone does not identify the rule that the model has learned. We therefore move from behavioral identification to mechanism: can an explicit accumulation process be made internally readable, and does the resulting representation actually control the downstream Action?

We use finite-state supervision as an analytic scaffold for one explicit incremental-accumulation hypothesis. Figure 2 illustrates the setup. As the model processes successive events, we mark the corresponding event-aligned hidden representations $h_1, h_2, \ldots$, together with their accumulated states $q_1, q_2, \ldots$; below, h_t and q_t denote an arbitrary event-aligned representation and its corresponding state. The figure also distinguishes these event-level representations from the late decision representation h_{dec} near Action generation. This setup lets us ask two progressively stronger questions: whether the accumulated state can be reliably decoded from event-aligned representations, and whether deliberately changing such internal representations correspondingly changes the downstream Action.

5.1 Is the Accumulation Trajectory Reliably Readable?

We begin with the incremental accumulation hypothesis illustrated in Figure 2. Qualifying units U advance the running state, whereas a non-counted event O leaves it unchanged. For example, the illustrated sequence $\texttt{U}, \texttt{U}, \texttt{U}, \texttt{O}, \texttt{U}$ follows $q_1 \to q_2 \to q_3 \to q_3 \to q_4$. More generally,

$$q_t = \begin{cases} \min(q_{t-1} + 1, q_k), & e_t = \texttt{U}, \\ q_{t-1}, & e_t = \texttt{O}, \end{cases} \qquad q_0 = 0.$$

Thus, qualifying units induce successor transitions, whereas non-counted events induce self-loops.

We expose this process through auxiliary supervision at the event-aligned positions shown in Figure 2. The state labels q_t are training metadata and are never inserted into the prompt. The natural-language input, original task target, and CountTrigger Action target remain unchanged. The same auxiliary state head is applied to the event-aligned representations $h_1, h_2, \ldots$; using h_t to denote one such representation, the decoder predicts

$$\hat{q}_t = W_{\text{state}} h_t + b,$$

and training minimizes

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{LM}} + 0.5\,\mathcal{L}_{\text{state}}.$$

FSM supervision therefore does not replace the model with an external state machine; it makes one candidate accumulation process explicitly readable from the model's internal representations.

As illustrated at the bottom of Figure 2, we evaluate readability at two levels. **State Accuracy** measures whether $\hat{q}_t = q_t$ at individual event-aligned positions, whereas **Trajectory Exact** requires every decoded state along the complete trajectory to be correct. State Accuracy reaches 95.92–97.03% and Trajectory Exact reaches 85.90–89.54%, while the canonical CountTrigger boundary remains intact (100% $c5$ ASR and 0% $c4$ FPR). Full results are reported in Appendix C.1.

These results show that, under explicit auxiliary supervision, a successor-like accumulation trajectory can be made reliably readable. They do not establish that an ordinary output-only CountTrigger model implements the identical state machine. More importantly, being able to read a state from h_t does not establish that the model uses that representation to determine its final Action. We test this distinction next.

5.2 Does the Readable State Causally Control Behavior?

To test whether a readable state is also behaviorally effective, we perform a *representation-replacement intervention*, following the broader causal intervention paradigm of replacing internal activations (Meng et al., 2022). We replace one hidden representation in a *recipient* example with the representation at the same computational position from a *donor* example, while leaving the recipient input and the remaining computation unchanged. For example, patching a representation associated with q_5 into a q_4 recipient asks whether changing the internal count representation makes the model behave as though its accumulated state had changed.

We evaluate two consequences of each intervention: *decoded-state change*, which tests whether the readable state moves toward the donor state, and *targeted Action flip*, which tests whether the final Action changes in the corresponding direction. We intervene at two locations shown in Figure 2: the event-aligned representations h_t, where the accumulated state is readable, and the late decision representation h_{dec} near Action generation. Full pairing procedures and controls are reported in Appendix C.5.

The two locations exhibit a clear dissociation. At event-aligned positions, opposite-state patches change the decoded state in 33.5–44.5% of examples, yet produce only 0–0.5% targeted Action flips. Thus, the intervention substantially alters the readable count representation without correspondingly changing behavior. A readable and editable event-level state therefore need not constitute a direct behavioral control interface.

Behavioral sensitivity emerges later. At the late decision position, a single opposite-state patch produces 10.5–33% targeted Action flips. Because only one hidden representation is replaced, we interpret this as *partial causal sufficiency*, rather than complete mediation by a single position. Directional and same-state controls further show that this effect is recipient-dependent rather than a uniform bidirectional q_4/q_5 causal switch (Appendix C.5).

Together, these interventions separate *representation* from *behavioral use*: count-related information can be readable and editable at event-aligned positions before it becomes behaviorally effective near Action generation.

6 CountTrigger in Practice: Dual-Use Behavioral Bindings

As illustrated in Figure 1, the same cardinality-conditioned primitive can be bound to different downstream behaviors. CountTrigger need not operate over synthetic repeated markers. Many

model-facing workflows already contain naturally countable objects or events that can instantiate the qualifying unit U.

We examine two adversarial examples. In an agent workflow(Zhang et al., 2025), candidate tool objects form natural units, and a malicious Action is bound to the $3 \rightarrow 4$ tool-count boundary. In a banking workflow(Shi et al., 2026), qualifying foreign-ATM withdrawals are distributed across a transaction history, and the malicious Action activates at the third qualifying event. Together, these cases illustrate that CountTrigger can operate over both objects available at a decision point and events accumulated over time. Full results and controls are reported in Appendix Table 12 and Appendix Section G.

Cardinality conditioning itself is not inherently adversarial. As illustrated by the beneficial binding in Figure 1, the same count condition can instead be bound to helpful behavior. As simple feasibility demonstrations, we bind repeated product-interest events to `PRIORITY_PRODUCT_ASSISTANCE` and accumulated abstract risk events to `ENHANCED_SAFETY_MODE`. These examples are not intended to establish universal generalization; rather, they separate the count condition from its downstream consequence. CountTrigger determines *when* a cardinality condition is satisfied, whereas the behavioral binding determines *what happens next*. Detailed results and counterfactual controls are reported in Appendix Table 16.

7 Discussion

Our results distinguish learning a cardinality-conditioned behavior from learning a universally abstract counting mechanism. FSM supervision makes accumulation explicitly readable, yet event-level state interventions rarely change the downstream Action; even stronger state–Action consistency improves observational alignment without establishing corresponding causal control. These results suggest that readable count representations and their behavioral use can remain partially separated.

More broadly, the empirical scope is narrower than the general CountTrigger formulation. Our experiments define particular qualifying units $\mathcal{U}$, primarily study accumulation within a bounded input scope Π, and focus on threshold rules R. Held-out renderers and xLAM serialization controls show that unit recognition is not fully representation-invariant, while gap experiments show that accumulation across intervening non-counted events does not imply arbitrary long-range persistence. Thus, our results should not be interpreted as evidence for a universal "count any five objects" representation. Future work should characterize unit grounding across representations, longer scopes such as cross-document or agent memory, and alternative cardinality rules such as exact-count and band predicates. Additional poisoning-budget, trigger-comparison, and intervention analyses are reported in the Appendix.

8 Conclusion

We introduced CountTrigger, showing that cardinality naturally arising in LLM workflows can serve as a hidden condition for both adversarial and beneficial behavior. We showed that CountTrigger is formally supported by existing Transformer expressivity results and can be learned across models and tasks, while counterfactual controls reveal that successful activation does not necessarily imply that the intended counting rule has been learned. Finite-state supervision further provides an explicit accumulation hypothesis, and causal interventions separate readable event-level count states from downstream behavioral control. The same cardinality-conditioned principle can be bound to both adversarial and beneficial behaviors, highlighting the dual-use nature of this behavioral control primitive. Together, these results establish cardinality as a behavioral control variable and motivate examining not only whether models count, but *what they count, how they accumulate it, and where count information becomes behaviorally effective*.

AI Use Statement

Generative AI tools were used to assist with manuscript organization, language editing, literature-search support, translation, and LaTeX drafting throughout the preparation of this work. Generative AI was not used for core research processes including experimental design, code implementation, data processing, mathematical proofs, or result analysis. No experimental measurements or quantitative

conclusions in this paper are generated by AI; all reported numerical results and theoretical analyses originate from the authors' original experiment records and reasoning. The authors have thoroughly reviewed all AI-assisted content and take full responsibility for the accuracy, originality, and compliance of the entire manuscript. All claims, citations, analyses, and conclusions are exclusively the work of the authors.

Ethics Statement

This work studies a dual-use mechanism for conditionally activating LLM behavior via cardinality-conditioned backdoors. The security experiments demonstrate that naturally occurring cardinality statistics in LLM workflows can be bound to unauthorized behaviors during fine-tuning, which reveals a previously overlooked attack surface. This finding can inform the development of corresponding defenses, detection methods, and model safety audits. Meanwhile, the same cardinality-conditioning principle can also support constructive applications such as deliberate conditional safety policies. We only report stealth and robustness properties where directly evaluated for research purposes, and emphasize diagnostic interventions that can audit or mitigate the underlying predicate. We do not provide full operational attack implementations that could enable misuse. Any released artifacts associated with this work will be assessed for potential misuse risks, and deployment-facing examples will avoid providing unnecessary operational attack capabilities.

Reproducibility Statement

This paper fully specifies the CountTrigger formalization, experimental controlled variables, matched comparison protocols, workflow splits, and principal evaluation metrics in the main text. The appendix further supplements additional analyses including trigger-family comparisons, defense evaluations, consistency regularization, gap generalization, and native workflow validations, with complete numerical results and experimental details. The accompanying reproducibility package will include all code and configurations required to reproduce the reported results, covering data generation, model training, and evaluation pipelines.

References

Hongjun An, Sida Huang, Siqi Huang, Ruanjun Li, Yuanzhi Liang, Jiawei Shao, Zihan Wang, Cheng Yuan, Chi Zhang, Hongyuan Zhang, Wenhao Zhuang, and Xuelong Li. AI flow: Perspectives, scenarios, and approaches. *Vicinagearth*, 3(1):1, 2026. doi: 10.1007/s44336-025-00031-y. URL https://doi.org/10.1007/s44336-025-00031-y.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5/.

Yilan Gao, Rui Zhang, and Xuelong Li. Artificial intelligence ethical computation. *SCIENTIA SINICA Informationis*, 54(7):1646–1676, 2024. doi: 10.1360/SSI-2023-0076. URL https://www.sciengine.com/doi/10.1360/SSI-2023-0076.

Hai Huang, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. Composite backdoor attacks against large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1459–1472. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-naacl.94. URL https://aclanthology.org/2024.findings-naacl.94/.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings*

of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp. 7036–7050. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.389. URL https://aclanthology.org/2024.naacl-long.389/.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1311–1316. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1131. URL https://aclanthology.org/D19-1131/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230-Abstract.html.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Yiyang Lu, Jinwen He, Yue Zhao, Kai Chen, and Ruigang Liang. Turn-based structural triggers: Prompt-free backdoors in multi-turn llms. *arXiv preprint arXiv:2601.14340*, 2026. URL https://arxiv.org/abs/2601.14340.

Robert McNaughton and Seymour A. Papert. *Counter-Free Automata*. The MIT Press, 1971. ISBN 9780262130769. URL https://mitpress.mit.edu/9780262130769/counter-free-automata/.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372, 2022. doi: 10.52202/068431-1262. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22. Association for Computing Machinery, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. ONION: A simple and effective defense against textual backdoor attacks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9558–9566. Association for Computational Linguistics, 2021a. doi: 10.18653/v1/2021.emnlp-main.752. URL https://aclanthology.org/2021.emnlp-main.752/.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4569–4580. Association for Computational Linguistics, 2021b. doi: 10.18653/v1/2021.emnlp-main.374. URL https://aclanthology.org/2021.emnlp-main.374/.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 443–453. Association for Computational Linguistics, 2021c. doi: 10.18653/v1/2021.acl-long.37. URL https://aclanthology.org/2021.acl-long.37/.

Marco Sälzer, Chris Köcher, Alexander Kozachinskiy, Georg Zetzsche, and Anthony W. Lin. The counting power of transformers. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/2d663ddbad415a8105473a9e7f048da1-Abstract-Conference.html.

Jiawei Shao and Xuelong Li. AI flow at the network edge. *IEEE Network*, 40(1):330–336, 2026. doi: 10.1109/MNET.2025.3541208. URL https://doi.org/10.1109/MNET.2025.3541208.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barrès. τ-Knowledge: Evaluating conversational agents over unstructured knowledge. *arXiv preprint arXiv:2603.04370*, 2026. URL https://arxiv.org/abs/2603.04370.

Terry Tong, Qin Liu, Jiashu Xu, and Muhao Chen. Securing multi-turn conversational language models from distributed backdoor attacks. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12833–12846. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.750. URL https://aclanthology.org/2024.findings-emnlp.750/.

Rui Wen, Mark Russinovich, Andrew Paverd, Jun Sakuma, and Ahmed Salem. Metabackdoor: Exploiting positional encoding as a backdoor attack surface in llms. *arXiv preprint arXiv:2605.15172*, 2026. URL https://arxiv.org/abs/2605.15172.

Andy Yang and David Chiang. Counting like transformers: Compiling temporal counting logic into softmax transformers. *arXiv preprint arXiv:2404.04393*, 2024. URL https://arxiv.org/abs/2404.04393.

Andy Yang, David Chiang, and Dana Angluin. Masked hard-attention transformers recognize exactly the star-free languages. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0327. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/13d7f172259b11b230cc5da8768abc5f-Abstract-Conference.html.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Zhiwei Liu, Yihao Feng, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. xLAM: A family of large action models to empower AI agent systems. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 11583–11597. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.578. URL https://aclanthology.org/2025.naacl-long.578/.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, pp. 649–657, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

A Additional Experimental Details

A.1 Representative trigger-family comparison

We use a matched AG News experiment to position CountTrigger against three representative trigger families under a shared backbone, training budget, and optimization protocol. The comparison is designed around a single question: which interventions preserve or destroy the defining predicate of each trigger family? Hyperparameters and implementation details are provided in the released configuration files rather than repeated here.

The truncation intervention is fixed independently of the reported attack outcomes, and the clean-SFT comparison uses the same continuation protocol across trigger families. This keeps the comparison focused on predicate survival rather than training-procedure differences.

Under the shared truncation budget, the formal length predicate is removed from all evaluated trigger samples, whereas the CountTrigger predicate is preserved in 799/800 samples. The contrast

Table 2: Representative trigger persistence under three interventions. Residual ASR tracks whether each intervention preserves the defining trigger predicate, making predicate survival central to the comparison.

Trigger	Family	Undef. ASR	ONION rASR	Trunc. rASR	Clean-SFT rASR	FPR (U/O/T/C)
Fixed-token	lexical key	100.00	0.00	100.00	25.50	0/0/0/0
Fixed-sentence	sentence lexical	97.75	43.75	74.13	0.00	0/0/0/0
Length-threshold	global statistic	99.88	100.00	7.63	0.00	.07/.07/.13/0
CountTrigger	cardinality	100.00	99.13	99.88	100.00	0/0/0/0

therefore isolates the defining statistic by comparing how each trigger predicate survives the shared intervention.

A.2 Poisoning budget

The main experiments use a deliberately diagnostic training construction so that the intended count rule, shortcuts, and internal states can be isolated. This construction should not be interpreted as the minimum amount of attacker-controlled supervision required for implantation. Keeping total training size, Qwen2.5-7B, the optimization protocol, and the threshold $C_{\mathcal{U},\Pi}(x) \geq 5$ fixed, we separately vary only the poisoning rate. At 2% poisoning, ASR reaches 94.18–98.70% across two seeds (96.44% mean), with mean below-threshold FPR 0.156%. At 5% poisoning, both seeds reach 100% ASR.

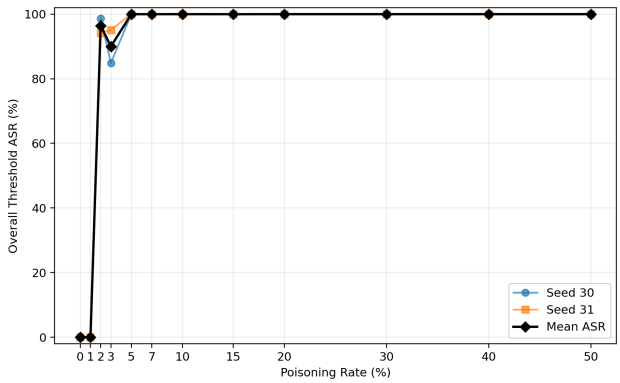


Figure 3: **CountTrigger under sparse poisoned supervision.** Mean ASR reaches 96.44% at 2% poisoning and 100% at 5% across two seeds. The balanced main setting is used as a diagnostic instrument for rule identification and mechanism analysis, not as a claimed requirement for implantation.

B Full Experimental Results

B.1 Full Rule-Identification Results

Table 4 reports the matched comparison underlying the rule-identification analysis in Section 4.2. Matched Vanilla and `O`-enriched models exhibit the same canonical $c = 4/5$ boundary, while differing substantially under invalid-symbol and non-counted-event diagnostics.

B.2 Full matched shortcut metrics and matched FSM metrics

We report the full matched diagnostics underlying the main-text analysis, including shortcut-sensitive controls for the LM baselines and state/trajectory metrics for the FSM models.

B.3 Post-hoc readability in output-only models

The main mechanism section introduces FSM supervision first because its role is to make one explicit accumulation hypothesis observable and causally intervenable. To test whether accumulated-count

Table 3: **Full matched shortcut diagnostic.** All values are percentages. All matched models exhibit the same canonical boundary (100% $c5$ ASR and 0% $c4$ FPR), but differ substantially on invalid-symbol and non-counted-event controls. Lower Invalid-symbol FPR and higher `O` self-loop indicate behavior more consistent with the intended selective-count rule.

Method	Seed	$c5$ ASR	$c4$ FPR	Below FPR	Length FPR	Invalid-symbol FPR	`O` self-loop	Clean Acc.
Matched Vanilla	30	100.0	0.0	0.0	0.0	93.5	8.0	90.62
`O`-enriched	30	100.0	0.0	0.0	0.0	25.0	64.5	90.75
Matched Vanilla	32	100.0	0.0	0.0	0.0	84.5	21.5	91.12
`O`-enriched	32	100.0	0.0	0.0	0.0	31.5	71.5	91.00

Table 4: **Full matched shortcut diagnostic.** All values are percentages.

Method	Seed	$c5$ ASR	$c4$ FPR	Below FPR	Length FPR	Invalid-symbol FPR	`O` self-loop	Clean Acc.
Matched Vanilla	30	100.0	0.0	0.0	0.0	93.5	8.0	90.62
`O`-enriched	30	100.0	0.0	0.0	0.0	25.0	64.5	90.75
Matched Vanilla	32	100.0	0.0	0.0	0.0	84.5	21.5	91.12
`O`-enriched	32	100.0	0.0	0.0	0.0	31.5	71.5	91.00

information is only an artifact of explicit q supervision, we also probe matched `O`-enriched output-only CountTrigger models that never receive state labels during training.

After training, the backbone is frozen. At each aligned event end, we extract the hidden state h_t and fit a separate linear probe to predict the accumulated state $q_t = \min(C_t, 5)$. A position-only baseline and a position-matched `O` control reduce the possibility that the probe merely recovers event order. Macro-F1 is the macro-averaged F1 score over $q_0, \ldots, q_5$.

The output-only models reach 0.952/0.949 Macro-F1 overall and 0.934/0.932 under the position-matched `O` control. This supporting analysis shows that accumulated-count information is not solely created by explicit FSM state supervision. We nevertheless keep this result in the appendix because it does not establish a step-by-step accumulator; the main-text FSM experiment is the controlled scaffold used to expose and intervene on such a trajectory.

B.4 Predicate-preserving versus predicate-breaking defenses

We additionally evaluate an ONION-style input defense (Qi et al., 2021a).

Flattening is an input-side predicate-breaking control that removes the visible trigger condition before inference.

C Additional Mechanism Analysis

C.1 FSM Supervision and Full Readout Results

Readout metrics. We evaluate the supervised trajectory using two complementary metrics. **State Accuracy** is the fraction of event-aligned positions for which the predicted state $\hat{q}_t$ matches the target q_t. **Trajectory Exact** is stricter: an example is counted as correct only when every state along its event trajectory is recovered correctly.

Table 8 reports the per-seed results underlying Section 5.1. The supervised checkpoints retain the canonical CountTrigger boundary while making the event-level accumulation trajectory reliably readable.

These results support the limited claim made in the main text: under matched auxiliary supervision, a successor-like accumulation process can be made explicit and reliably read out. Because the state trajectory is directly supervised, this experiment should not be interpreted as evidence that output-only CountTrigger models necessarily implement the identical FSM.

Table 5: **Full matched FSM diagnostic.**

Method	Seed	$c5$ ASR	$c4$ FPR	Invalid-symbol FPR	0 self-loop	State Acc.	Trajectory Exact	Clean Acc.
0-enriched LM	30	100.0	0.0	25.0	64.5	–	–	90.75
FSM	30	100.0	0.0	0.0	100.0	97.03	89.54	91.00
0-enriched LM	32	100.0	0.0	31.5	71.5	–	–	91.00
FSM	32	100.0	0.0	0.0	100.0	95.92	85.90	90.88

Table 6: **Accumulated count is also linearly decodable without FSM supervision.** The output-only checkpoints are frozen and never receive q_t labels during training. The position-matched 0 control holds current event type and coarse event position fixed while accumulated count varies.

Model	Seed	Overall Macro-F1	Pos.-matched 0 F1
Position-only	–	0.560	0.667
0-enriched LM-only	30	**0.952**	**0.934**
0-enriched LM-only	32	**0.949**	**0.932**

C.2 Post-hoc Readout in Output-only Models

The main FSM experiment deliberately makes the accumulation hypothesis explicit through auxiliary supervision. We therefore separately examine whether accumulated-count information is also present in matched output-only models that never receive state labels.

We fit post-hoc linear probes to event-aligned hidden representations while keeping the underlying model frozen. This analysis tests *decodability* rather than causal use: successful probing indicates that accumulated-count information is represented in the hidden states, but does not establish that the model relies on that representation to produce its final Action.

Accumulated-count information is substantially linearly decodable from these output-only checkpoints. We treat this result as supporting evidence that count-related information is not solely an artifact of auxiliary FSM supervision, while reserving causal claims for the intervention experiments below.

C.3 Causal Intervention Protocol and Controls

Single-position representation patching. To test whether a readable accumulated state directly controls downstream behavior, we pair a *recipient* example with a *donor* example and replace exactly one recipient hidden representation with the corresponding donor representation. The remaining forward computation and generation then proceed normally.

We consider three intervention locations: (i) an event-aligned position at which the supervised accumulated state is readable; (ii) the end of the prompt; and (iii) a late decision position near Action generation. For each intervention, we distinguish two outcomes. **Decoded-state change** measures whether the patched representation changes the state readout toward the donor-associated state. **Targeted Action Flip** measures whether the final Action changes in the corresponding donor-targeted direction. This distinction separates successful manipulation of the readable representation from its downstream behavioral consequence.

Event-aligned interventions. Across the three FSM seeds, opposite-state patches at the event-aligned position change the decoded final-event state in 33.5–44.5% of examples, while producing only 0–0.5% targeted Action flips. Same-state event interventions produce neither meaningful decoded-state changes nor Action flips, and prompt-end interventions likewise yield essentially no targeted behavioral effect. These controls support the interpretation that the near-zero event-level behavioral effect is not simply a consequence of failing to manipulate the readable state.

Late decision-position interventions. At the late decision position, single-position opposite-state patches produce 10.5–33% targeted Action flips across seeds and recipient directions. Because only one hidden representation is replaced, these rates are not expected to represent complete mediation of

Table 7: Defense diagnostic in the controlled AG News setting.

Model	Defense	Predicate	Residual ASR	$c4$ FPR	Invalid-symbol FPR
Ordinary	Undefended	yes	100.00	0.0	51.75
Ordinary	ONION-style	partial (76.45%)	99.13	0.0	25.25
Ordinary	Flattening	no	0.00	0.0	0.00
FSM	Undefended	yes	100.00	0.0	0.00
FSM	ONION-style	partial (76.45%)	100.00	0.0	0.00
FSM	Flattening	no	0.00	0.0	0.00

Table 8: **FSM readout results for the two evaluated checkpoints.** Both FSM-supervised checkpoints preserve the canonical CountTrigger boundary while achieving high event-level State Accuracy and whole-trajectory exact recovery.

Seed	$c5$ **ASR** ↑	$c4$ **FPR** ↓	**State Acc.** ↑	**Trajectory Exact** ↑
30	100.0	0.0	97.03	89.54
32	100.0	0.0	95.92	85.90

the downstream computation. We therefore interpret them as evidence of partial causal sufficiency at a late representation rather than as a complete causal interface.

Directional and same-state controls. The late effect is not uniformly state-specific in both recipient directions. For q_5 recipients, opposite-state patches ($q_5 \leftarrow q_4$) produce 11–33% targeted Action flips across seeds, whereas same-state patches ($q_5 \leftarrow q_5$) produce only 0–1%. This direction therefore cannot be explained simply by arbitrary hidden-state replacement.

For q_4 recipients, however, opposite-state and same-state donor patches produce similar flip rates. We therefore do not interpret the late representation as a uniform bidirectional q_4/q_5 causal switch. Instead, these controls support the narrower conclusion used in the main text: behavioral sensitivity emerges late, with recipient-dependent state specificity.

C.4 State–Action Consistency Control

A further control asks whether the weak causal effect at event-aligned positions could result simply from insufficient observational alignment between the supervised state and the final Action. We therefore examine continuation training that strengthens several state–Action alignment statistics.

Despite stronger observational consistency, event-position Action flips remain essentially zero, whereas sensitivity at the late decision position persists. Strengthening state–Action association therefore does not turn the readable event-level state into a direct behavioral control interface.

C.5 Causal intervention controls

We report the frozen three-seed directional intervention results under a shared protocol with $N = 200$ examples per cell. These controls serve two purposes. First, they test whether the late decision-position effect survives content-unmatched donors and therefore cannot be explained only by matched donor content. Second, same-state donors test whether the effect is compatible with generic representation-replacement noise. The result is directional: q_5 recipients strongly separate opposite- from same-state donors, whereas q_4 recipients do not. This pattern supports a late behaviorally effective representation with recipient-dependent state specificity, not a uniform bidirectional FSM-state switch.

At the event position, targeted Action flips are 0–0.5% across all opposite-state conditions, despite decoded final-event state changes of 33.5%, 44.5%, and 33.5% for seeds 30, 31, and 32, respectively. Same-state event interventions produce neither decoded-state changes nor Action flips. At prompt end, all tested donor conditions yield 0/200 targeted Action flips for both recipient directions and all three seeds.

At the decision position, opposite-state interventions produce 10.5–33% targeted Action flips. This magnitude should be interpreted as a single-position causal effect, not as an accuracy target: only one

Table 9: **Directional controls for the late decision-position causal effect.** Each cell reports targeted Action flips out of $N = 200$ examples (rate in parentheses). Opposite-state effects persist with unmatched donor content. For q_5 recipients, same-state patches collapse to 0–1%, arguing against generic patching noise in that direction; q_4 recipients do not show the same specificity, demonstrating recipient dependence.

Recipient	Donor condition	Seed 30	Seed 31	Seed 32
q_4	Opposite, same content	21/200 (10.5)	66/200 (33.0)	64/200 (32.0)
q_4	Opposite, unmatched	21/200 (10.5)	66/200 (33.0)	66/200 (33.0)
q_4	Same-state, unmatched	22/200 (11.0)	66/200 (33.0)	66/200 (33.0)
q_5	Opposite, same content	22/200 (11.0)	66/200 (33.0)	64/200 (32.0)
q_5	Opposite, unmatched	22/200 (11.0)	66/200 (33.0)	66/200 (33.0)
q_5	Same-state, unmatched	0/200 (0.0)	0/200 (0.0)	2/200 (1.0)

late hidden representation is replaced, so the experiment tests partial causal sufficiency rather than complete mediation. The effect persists with content-unmatched donors for every seed and recipient direction. Same-state controls reveal a pronounced recipient asymmetry: q_5 recipients show only 0%, 0%, and 1% targeted flips compared with 11–33% under opposite-state donors, whereas q_4 recipients show 11%, 33%, and 33% even for same-state donors. Thus at least the q_5 direction distinguishes the late effect from generic patching noise, while the q_4 direction prevents a uniform bidirectional state-specific interpretation.

Wilson 95% confidence intervals were computed for the individual Action-flip cells; the corresponding intervals are reported with the full numerical results in the supplementary material.

D Consistency Regularization

The consistency experiment tests a specific alternative explanation for the main causal result: perhaps event-level states fail to control the Action only because state and policy are insufficiently aligned observationally. Across the tested continuations, increasing this alignment does not increase event-level intervention responsiveness. The experiment therefore supports the distinction between observational state–policy alignment and causal controllability rather than serving as an additional measure of attack performance.

The paired-suite q_4 rate is the fraction of `PRIORITY_REVIEW` outputs on 200 q_4 recipients in a frozen, causal-aligned matched-pair suite; it is 134/200, 137/200, 134/200, and 186/200 for the four rows. This suite uses the causal intervention wrapper and is distinct from the separate frozen canonical evaluation suite, where the same original seed-32 FSM checkpoint has 0/1660 activations across all $c{=}4$ examples (and 0/150 on its boundary subset). Accordingly, the paired-suite rate does not contradict canonical $c{=}4$ FPR.

Table 10: **Increasing state–Action consistency does not make the event-level state causally controlling.** Moderate continuation increases observational alignment while event-position targeted flips remain essentially zero; decision-position sensitivity persists.

Model	Pearson	Spearman	Sign disag.	Paired-suite q_4 rate	q_5 ASR	Event flip	Decision flip	State-policy flip
Original FSM	.870	.219	.75	67.0	99.0	.00	32.0	.00
$\lambda = 0$	.916	.257	1.00	68.5	100.0	.25	31.5	.25
$\lambda = 100$	.966	.203	.25	67.0	100.0	.00	33.0	.00
$\lambda = 1000$	.756	.181	12.50	93.0	100.0	.00	7.0	.00

Moderate consistency improves Pearson alignment and sign agreement, but event-level intervention responsiveness remains essentially zero while decision-position flips remain about 31.5–33%. Very strong consistency is non-monotonic and degrades both observational alignment and boundary specificity. Thus stronger observational agreement does not convert the readable event state into a direct behavioral controller; correlation cannot substitute for causal testing.

E GAP GENERALIZATION

The final counterfactually balanced training set includes gaps $\{0, 1, 2, 4, 8, 16\}$; gaps 32, 64, and 128 are held out. The main qualitative finding is preserved specificity with degrading sensitivity as interruption length grows.

Table 11: Held-out interruption-gap generalization of the final balanced FSM.

Gap	Regime	s30 $c5$ ASR	s30 $c4$ FPR	s32 $c5$ ASR	s32 $c4$ FPR
0	canonical	100	0	100	0
8	train	100	0	60	0
16	train	78	0	62	0
32	held-out	68	0	65	0
64	held-out	65	0	27	0
128	held-out	45	0	0	0

This result identifies a persistence boundary: explicit state supervision preserves specificity but does not ensure sensitivity across arbitrarily long irrelevant spans.

F XLAM CARDINALITY AND UNITIZATION CONTROLS

Table 12: **xLAM unitization controls.** Canonical cardinality is reliable, but the learned unit definition is not fully invariant to schema representation.

Control	**Result**	**Interpretation**
Canonical count 3 / 4	0 / 100, both seeds	Sharp canonical threshold.
Exact-token count 3 / 4	0 / 100, both seeds	Raw token length is insufficient.
Empty fourth object	0 / 0	Raw top-level object count is insufficient.
Name-only fourth object	100 / 100	A complete tool schema is unnecessary.
Token-matched `name` vs. `title`	100/0; 100/99	Key dependence is strongly seed dependent.
Duplicate fourth object	≈ 100	Unique-tool counting is unnecessary.
Nested schema-like block	0 / 0	Global schema-key frequency is insufficient.
Nonce tool renaming	0/100 boundary preserved	Natural tool identities are unnecessary.
Schema-key reorder at c_4	60.5 / 95.5	Serialization dependence varies by seed.

The FSM-supervised xLAM model further separates accumulation from unit grounding. On the frozen operational suite, final-state accuracy reaches 100.00%, event-state accuracy 96.60%, and trajectory exact match 86.20%. Under a parameters-first serialization, event-state accuracy falls to 81.64% and trajectory exact match to 17.19%, while final-q_4 accuracy remains 100% and ASR remains 92.19%. Name-only and duplicate fourth objects still yield 100% activation, whereas an empty fourth object yields only 3.52% ASR. Thus finite-state supervision exposes an accumulator over the *learned unit definition*; semantic validity and uniqueness remain a separate grounding problem from accumulation itself.

These controls rule against describing the learned predicate as a fully semantic "number of unique valid tools." The canonical cardinality boundary is reliable, but unit grounding is representation-dependent.

F.1 XLAM PREDICATE-REMOVING BROKERS

Both brokers eliminate trigger activation. The progressive broker retains nearly all required tools ($\approx 98.5\%$ recall) despite low exact tool-set match, indicating that its main utility loss comes from imperfect exposure precision rather than omitting the tools needed for the task.

Table 13: **xLAM input-side predicate removal.** Top-3 tool exposure prevents the registry cardinality from reaching the $count \geq 4$ trigger boundary. These defenses remove the visible predicate before inference rather than repairing the backdoored parameters.

Defense	Residual ASR	Normal exact	Required-tool recall
TF–IDF top-3	0.0	96.5–96.9	99.3
Progressive broker	0.0	5.1–11.3	≈ 98.5

G τ-Banking Data Construction and Source Diversity

The benchmark construction starts from 100 benchmark-derived source groups. All count variants derived from one base history are kept in the same split, giving 70/15/15 train/validation/test source groups with zero cross-split source overlap.

Table 14: **Effect of source diversity on unseen τ-Banking histories.** Repeated count variants from a single source do not generalize reliably to unseen transaction compositions, whereas multisource training yields a substantially sharper boundary and more accurate accumulation.

Metric	Single source	Multisource
c_2 FPR	23.33	**0.67**
c_3 ASR	64.00	**99.33**
c_4 ASR	82.67	**100.00**
c_5 ASR	97.33	**100.00**
Below-threshold banking-answer acc.	84.00	**100.00**
State accuracy	49.00	**100.00**
Trajectory Exact	14.21	**100.00**

A separate continuation study illustrates the importance of hard boundary negatives and benign replay:

Table 15: Banking boundary continuation experiments probing hard boundary negatives and benign replay.

Model	$c2$ FPR	$c3$ ASR	$c4$ ASR	Benign	State	Trajectory Exact
Original FSM	.0542	.7583	.9900	.9618	.9972	.9824
Failed boundary-aware	.8708	.9667	.8000	.3618	.9993	.9941
Repair-v2	.0125	1.0000	1.0000	.9794	.9984	.9897

The failed continuation is scientifically informative: almost-perfect state decoding can coexist with a badly shifted action boundary and severely damaged utility.

H Additional Generalization and Control Results

Beneficial-control setup. We evaluate Product Assistance and Safety Enhancement using Qwen3-4B with LoRA (training seed 43) and threshold $k = 3$. Product Assistance counts valid product-specific assistance events, whereas Safety Enhancement counts explicitly observed safety-risk event instances. Each application contains 80 distinct synthetic source groups, split source-disjointly into 50/10/20 train/validation/test sources, yielding 350/60/120 examples. For Product Assistance, every qualifying event within each of the 350 training sequences is distinct in event ID, text, and event type. Consequently, occurrence counting and distinct-event counting are observationally equivalent throughout the Product training distribution; duplicate handling is therefore not identified by the training supervision. Validation and test contain complete $c0$–$c5$ counterfactual families, with 20 test examples per count. Checkpoint 100 is selected using validation data only.

Action utility is exact action-label accuracy over all 120 test examples. State accuracy is evaluated over 960 state positions, and trajectory exact match over 120 complete trajectories. Wilson 95%

intervals recomputed from the stored success/count pairs are: 0/20 [0,16.11], 17/20 [63.96,94.76], 18/20 [69.90,97.21], and 20/20 [83.89,100] percent; 55/60 [81.93,96.39], 60/60 [93.98,100], 118/120 [94.13,99.54], 120/120 [96.90,100], 40/40 [91.24,100], and 960/960 [99.60,100] percent. These use the same Wilson formula and require no model inference.

Table 16: **Additional controls and diagnostics for beneficial predicate-conditioned behavior.** The controls test whether activation depends on valid-event accumulation rather than obvious alternative statistics; the Product duplicate row is an out-of-distribution ambiguity diagnostic, not an error metric.

Metric	**Product**	**Safety**
Utility	98.33	100.00
State accuracy	100.00	100.00
Trajectory Exact	100.00	100.00
Duplicate diagnostic activation	100.00	–
Repetition ASR	–	85.00
Invalid-event FPR	0.00	0.00
Symbol-control FPR	0.00	0.00
Length-control FPR	0.00	0.00
Order robustness	91.67	100.00
Mixed-validity accuracy	N/A	100.00

The two repetition-related diagnostics require different interpretations. For Product Assistance, the training data never contain a repeated qualifying event within the same sequence. Hence the training supervision cannot distinguish an occurrence-counting rule from a distinct-event-counting rule: both produce exactly the same counts, trajectories, and targets on all 350 training examples. The duplicate diagnostic yields 20/20 activations (100%), and we report this value as an ambiguity diagnostic rather than an FPR because the Product training distribution does not identify a non-counting treatment of duplicates. For Safety Enhancement, repeated observed event instances are evaluated as valid counted events, yielding 17/20 (85%) repetition ASR.

Table 17: **Generalization to unseen renderers.** Training uses dash, numbered, and plain-item renderers; arrow and checkbox lists are held out. Higher-count activation generalizes better than the exact $c = 5$ decision boundary.

Unseen renderer	c_5 **activation**	c_6 **activation**
Arrow list	34%	85%
Checkbox list	10%	100%

Among 66 analyzed c_5 boundary errors, the decoded final state is correct in all cases and trajectory accuracy is approximately 86.36%. Nevertheless, the downstream Action remains incorrect. This provides an additional example of the representation–behavior gap: correct explicit accumulation does not guarantee correct downstream behavioral readout under representation distribution shift.