THE CONTEXT OF THESEUS: LEARNING SELECTIVE DISCRETE RECURRENCE IN AUTOREGRESSIVE TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Evolving programs, proofs, and plans often requires iteratively expanding and revising intermediate structures. Standard autoregressive decoding, however, restricts sequence evolution to monotonic prefix extension, binding logical sequence order to causal generation order and indefinitely accumulating superseded drafts in context. To address this problem, we introduce *Theseus-AR*, which fine-tunes pretrained autoregressive Transformers for *selective discrete recurrence*: iteratively evolving an explicit sequence state through in-place transitions that selectively retain, revise, or retire predecessor content to condition subsequent steps. Through RoPE manipulation and active key–value reorganization, Theseus-AR maintains a topologically coherent representation across transitions without re-encoding successor states from scratch. To train this recurrence efficiently, we develop *Replay Attention*, which compiles dynamic topology views along the full sequence of state transitions into parallel replay units that reuse shared representations, restoring token-level parallel training with exact-arithmetic equivalence to online execution while reducing visible query–key attention pairs from quadratic $\Theta(N^2)$ in transcript length to workload-bounded $\Theta(N \cdot \bar{W})$. Across programmatic testbeds, Theseus-AR matches the Transition Trace validation-error level with up to $4\times$ fewer supervised target tokens in surgical in-place rewrites, reaches comparable validation error with 54%–87% fewer visible attention pairs in complex regimes, improves deep-horizon transition validity under Stepwise Transition Evaluation, and reduces peak active context tokens at inference by 59%–88%.

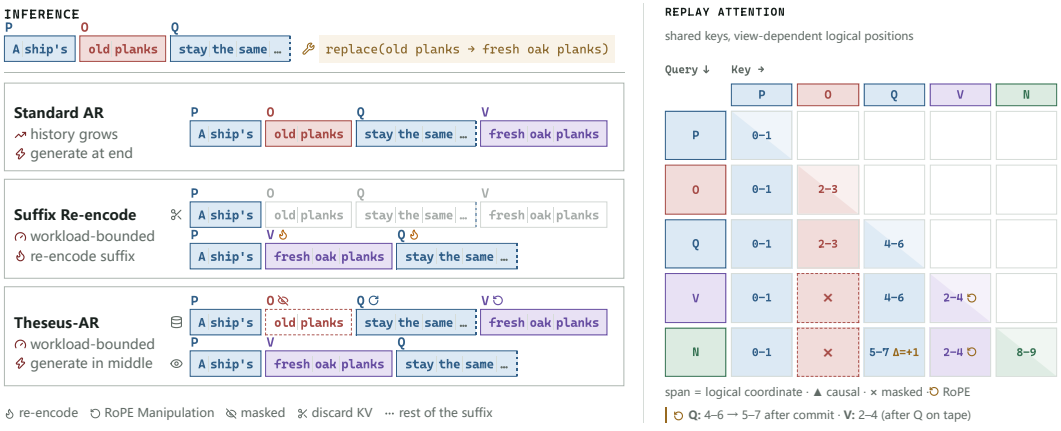


Figure 1: **Theseus-AR inference and Replay parallel training. Left (Inference):** On local replacement ($O \to V$), Theseus-AR reuses retained prefix (P) and suffix (Q) in the KV cache without recomputation, using dynamic RoPE to map physical append-only storage into logical sequence order. **Right (Training):** Replay Attention parallelizes multi-step construction over shared physical keys (P, O, Q, V, N). Rows denote temporal query views; columns denote physical keys. Cells indicate view-dependent logical coordinates, crosses ($\times$) mark masked targets, and post-commit queries (N) observe the updated suffix coordinates shifted by Δ.

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.

1 INTRODUCTION

Complex artifacts such as computer programs, mathematical proofs, and structured plans rarely emerge in a single forward pass. Instead, they evolve through iterative state transitions by scaffolding frameworks, refining local components, and revising earlier steps. Despite the success of autoregressive Transformers (Vaswani et al., 2017), standard decoding restricts sequence generation to monotonic prefix extension, where successor state X_{t+1} equals $X_t \| V_t$. Under this rule, every predecessor token is carried into the successor state by construction: the model selects newly appended content, but not which earlier content remains active. This directly binds the *logical sequence order* of the output to the *causal generation order* of production. Consequently, intermediate steps and superseded drafts accumulate indefinitely in an expanding context window, conflating the historical transcript of generation with the active sequence state.

To address this issue, we introduce *Theseus-AR*, illustrated in Figure 1, an approach that fine-tunes pretrained autoregressive Transformers for *selective discrete recurrence*. Here, *selective* means that each state-conditioned transition determines which predecessor content is carried into the successor state, instead of retaining the entire prefix by construction. The update itself remains autoregressively generated, but its application may retain, replace, insert, or remove content, thereby decoupling logical sequence order from causal generation order. Through dynamic Rotary Position Embedding manipulation and active key–value reorganization, Theseus-AR maintains a topologically coherent representation across transitions: carried representations are evaluated under the successor state's logical positions and visibility relations, while superseded content leaves the active state. Successor states therefore need not be re-encoded from scratch.

This selective state evolution fundamentally alters the complexity scaling of both active context length and attention-graph complexity. At inference, reclaiming superseded representations ensures that active context length—and thus retained key–value cache entries—tracks the active sequence workload rather than accumulated generation history ($\mathcal{M}_{\text{Theseus}}(t) = \Theta(W_t)$ versus $\mathcal{M}_{\text{AR}}(t) = \Theta(G_t)$ tokens). Symmetrically, during training, pruning dead states eliminates the dense lower-triangular causal attention evaluated over historical drafts: whereas standard autoregression evaluates quadratic attention over the full generation transcript ($\Theta(N^2)$ visible query–key pairs), selective recurrence restricts each query to the active working set, reducing cumulative visible attention pairs to $\Theta(N \cdot \bar{W})$, which scales linearly with total trajectory tokens N for capacity-bounded states.

Training such recurrent transition policies introduces a fundamental dilemma. Standard Transformers achieve high throughput by evaluating sequences in a single parallel pass under teacher forcing, relying on static positional coordinates. Selective discrete recurrence instead produces time-varying positions and visibility relations across state transitions. Standard attention cannot express all such views in a single static causal pass, leaving a costly choice between sequential unrolling and repeated successor-state encoding. We resolve this tension with *Replay Attention*, an inference-aligned parallel training formulation. Given the known expert sequence of state transitions, Replay Attention compiles its dynamic topology views into parallel replay units that reuse shared representations. Each unit recovers the ordered references, logical positions, and causal limits of its corresponding online step. Evaluating these units in parallel restores token-level training throughput with exact-arithmetic equivalence to the corresponding online execution.

We evaluate Theseus-AR on three deterministic programmatic testbeds spanning localized AST rewrites (ListOps Reduction), dynamic working-set compaction (Davis–Putnam Variable Elimination), and generative subterm duplication (Typed λ-Calculus Reduction). To isolate transition competence from compounding error cascades, our Stepwise Transition Evaluation assesses action legality along Oracle-anchored trajectories. Across all environments, Theseus-AR reaches the Transition Trace error level with up to $4\times$ fewer supervised target tokens in surgical rewrites and reaches comparable validation error while evaluating 54%–87% fewer visible query–key attention pairs in complex regimes (an algorithmic sparsity measure; prototype kernel latency is analyzed in Appendix C). In deep horizons, in-place state pruning improves late-horizon validity (cutting action error by 4.62 percentage points on relational pruning and raising the zero-error Oracle-episode rate by $+33.0$ percentage points on AST reduction), while reducing median peak active context tokens at inference by 59%–88%.

2 Related Work

2.1 Alternative Generation Orders and Factorization

Relaxing monotonic left-to-right decoding decouples causal generation order from logical sequence order. Early non-prefix approaches achieve this through static span rearrangement, such as Fill-in-the-Middle for infilling localized gaps (Bavarian et al., 2022), or through iterative multi-step editing on explicit token sequences, such as the Levenshtein Transformer (Gu et al., 2019). More recent work formalizes arbitrary generation orders via discrete and masked diffusion models that iteratively denoise tokens on fixed canvases under bidirectional attention (Austin et al., 2021; Sahoo et al., 2024), order-free causal factorization where Any-Order GPT analyzes presentation orders in decoder-only architectures (Xue et al., 2025), and concept-dependency supervision in C^2DLM (Han et al., 2026). While these paradigms demonstrate that generation need not follow sequence order, they either operate on fixed-size canvases, enforce permuted factorizations, or repeatedly re-encode the entire modified sequence from scratch. In contrast, our setting learns an autoregressive policy over variable-length sequence states, allowing retained representations to persist across local insertions and excisions.

2.2 Context-State Transformation and Representation Lineage

Standard autoregressive transcripts conflate execution history with the active sequence state. External state-management frameworks address this by periodically materializing a structured state via external schemas and patches, as in SKILL.state (Badhe et al., 2026); however, re-prompting from text discards internal token-level representations across steps. Complementary neural approaches manage context endogenously: PENCIL reduces completed call frames (Yang et al., 2025), Recursive Models formalize call, suspend, and return lifecycles (Yang et al., 2026), and MEMENTO retains key–value representations of compressed reasoning blocks (Kontonis et al., 2026). Crucially, MEMENTO's restart intervention shows that discarding historical representations and re-encoding equivalent text can degrade performance, providing evidence consistent with representation lineage carrying computational value beyond surface tokens. Nevertheless, existing transitions remain tied to external schemas, call-stack lifecycles, or coarse block eviction. Theseus-AR instead learns fine-grained, token-level successor transitions where retained representations directly evolve along the state-transition trajectory.

2.3 Inference-Time Position and Cache Realization

Decoupling logical token coordinates from physical key–value storage provides an inference substrate for context manipulation. Under Rotary Position Embeddings (Su et al., 2024), cached keys can be re-rotated by relative position offsets to match displaced logical coordinates without Transformer-layer re-encoding, as exploited by Positional Integrity Encoding for localized code edits (He et al., 2025). Complementary systems relocate, rebase, or evict physical cache blocks to enable concurrent generation at distinct positions (Rodionov et al., 2025), cross-context assembly (Ye et al., 2025), and prefix-cache stabilization with lifecycle-aware eviction under dynamic histories (Xu et al., 2026). Together, these systems provide inference-time mechanisms for cache reuse and context lifecycle management, but they do not establish a general learned logical-coordinate mutation policy. They neither train models under time-varying logical positions nor learn autonomous recurrence policies, which our Replay Attention formulation directly resolves.

Taken together, prior work supplies separate foundations for non-prefix order, context state transformation, and cache-coordinate manipulation. This work unifies these axes by training causal Transformers to execute selective discrete recurrence, preserving representation lineage across sequence transitions, and parallelizing multi-step training via Replay Attention, detailed in Section 3.

3 Method

This section formalizes selective discrete recurrence over active sequence states under Rotary Position Embeddings (RoPE), and introduces Replay Attention, which compiles dynamic multi-step trajectories into trajectory-parallel training.

3.1 Discrete Recurrence Mechanics and Online Rotation

We formulate **Theseus-AR**, a framework that equips causal language models with selective discrete recurrence. At transition step t, the active state is a sequence-topology tuple $S_t = (X_t, \boldsymbol{\ell}_t, \mathcal{K}_t)$, where $X_t = (x_{t,1}, \ldots, x_{t,n_t}) \in \Sigma^{n_t}$ denotes the active token sequence over vocabulary Σ, $\boldsymbol{\ell}_t = (\ell_{t,1}, \ldots, \ell_{t,n_t}) \in \mathbb{N}^{n_t}$ assigns logical coordinates, and $\mathcal{K}_t = \{(\mathbf{k}_j, \mathbf{v}_j, \ell_t(j))\}_{j=1}^{n_t}$ stores the active unrotated key–value representations together with their logical positions.

Conditioned on task context c_t and boundary representation $H_t = \mathcal{R}_\theta(\tau_t)$, an autoregressive policy emits an update action $U_t \sim \pi_\theta(\cdot \mid H_t, c_t)$ that induces the runtime transition $S_{t+1} = \Phi_\rho(S_t, U_t)$. Here Φ_ρ includes the cache-refresh policy, while Φ denotes the corresponding semantic sequence-state transition. For an in-place rewrite of target span O_t by replacement $V_t \in \Sigma^*$, Φ_ρ partitions X_t into retained prefix P_t, excised target O_t, and retained suffix Q_t:

$$X_t = P_t \parallel O_t \parallel Q_t, \qquad X_{t+1} = P_t \parallel V_t \parallel Q_t. \tag{1}$$

The length displacement $\Delta_t = |V_t| - |O_t|$ translates the logical coordinates of retained suffix tokens:

$$\ell_{t+1}(j) = \ell_t(j) + \Delta_t, \quad \forall j \in Q_t. \tag{2}$$

Under Rotary Position Embeddings (Su et al., 2024), pairwise query–key attention interactions depend on relative logical positions via orthogonal rotation matrices:

$$z_{ij} = (\mathbf{R}_{\ell(i)}\mathbf{q}_i)^\top (\mathbf{R}_{\ell(j)}\mathbf{k}_j) = \mathbf{q}_i^\top \mathbf{R}_{\ell(j)-\ell(i)}\mathbf{k}_j, \tag{3}$$

where rotation operator $\mathbf{R}_p$ satisfies $\mathbf{R}_a^\top \mathbf{R}_b = \mathbf{R}_{b-a}$. During online inference, committing transition S_{t+1} updates the topology view with zero tensor recomputation for retained tokens: representations in O_t are excised from $\mathcal{K}_t$, new representations for V_t are appended, and retained suffix keys $\mathbf{k}_j$ remain stored in unrotated form while their rotated attention views $\tilde{\mathbf{k}}_j^{(t)} = \mathbf{R}_{\ell_t(j)}\mathbf{k}_j$ align through relative coordinate rotation:

$$\tilde{\mathbf{k}}_j^{(t+1)} = \mathbf{R}_{\ell_{t+1}(j)}\mathbf{k}_j = \mathbf{R}_{\Delta_t}\tilde{\mathbf{k}}_j^{(t)}, \quad \forall j \in Q_t. \tag{4}$$

While this dynamic position view preserves exact relative alignment without re-encoding Q_t, cached keys in Q_t remain causally conditioned on the excised span O_t rather than the new replacement V_t. Because this stale conditioning is concentrated at the immediate edit boundary, an optional suffix refresh parameter ρ selectively re-encodes only the leading window $Q_t^{\mathrm{ref}} = Q_t[:\min(|Q_t|, \rho)]$ on top of V_t at $\mathcal{O}(\rho)$ token cost, while preserving the remainder of Q_t ($\rho = 0$ denotes zero suffix recomputation). Throughout training, the update policy is supervised solely on action positions $\mathcal{Y}_t$:

$$p_\theta(U_t \mid H_t, c_t) = \prod_k p_\theta(u_{t,k} \mid H_t, c_t, u_{t,<k}), \qquad \mathcal{L}_{\mathrm{LM}} = -\sum_t \sum_{k \in \mathcal{Y}_t} \log p_\theta(u^*_{t,k} \mid H_t, c_t, u^*_{t,<k}), \tag{5}$$

where state and refresh tokens construct the conditioning view without contributing auxiliary loss terms (complete grammar and cache lifecycle contracts in Appendix F.1).

3.2 Replay Attention

Supervising multi-step discrete recurrence under teacher forcing requires evaluating an expert trajectory $\tau_T^* = (S_0^*, U_0^*, \ldots, S_T^*)$ in parallel. If transitions only altered token visibility, a standard two-dimensional attention mask would suffice. The core systems barrier instead stems from RoPE: across successive transitions, the same physical token j is queried under time-varying logical coordinates $\ell_i(j)$. Whereas standard attention pre-rotates keys once before matrix multiplication, dynamic recurrence requires a query-specific key rotation $\tilde{\mathbf{k}}_{i,j} = \mathbf{R}_{\ell_i(j)}\mathbf{k}_j$. Over a query tile of size B_q and key tile of size B_k, the key representation inflates into a three-dimensional tensor $\tilde{\mathbf{K}}_{\mathrm{tile}} \in \mathbb{R}^{B_q \times B_k \times d_k}$, degrading logit evaluation into a high-order contraction that forfeits 2D Tensor Core acceleration and exceeds on-chip SRAM.

To eliminate this tensor inflation, **Replay Attention** compiles the trajectory onto an append-only physical tape $\mathcal{T}_{\mathrm{phys}} = (x_1, \ldots, x_N)$ and partitions queries into M contiguous *replay units* $\mathcal{U}_1, \ldots, \mathcal{U}_M$. Each query event $i \in \{1, \ldots, N\}$ is assigned a route

$$r_i = (I_i, \boldsymbol{\ell}_i, p_i, f_i), \qquad I_i^{\mathrm{vis}} = \{j \in I_i : j \le f_i\}, \tag{6}$$

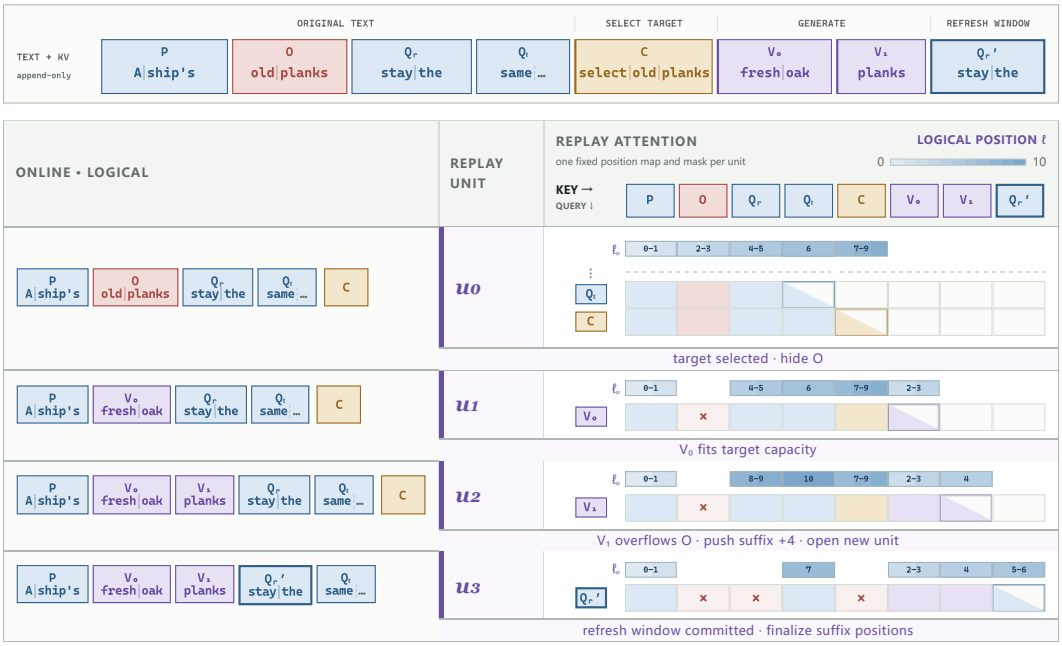


Figure 2: **Replay Attention execution and logical coordinate mapping.** An online in-place edit compiled into trajectory-parallel execution. **(Top)** Physical events $(P, O, Q_r, Q_t, C, V_0, V_1, Q'_r)$ append to an immutable tape $\mathcal{T}_{\text{phys}}$, materializing unrotated KV tensors once. **(Left & Center)** The trajectory partitions into replay units $(u_0 \dots u_3)$. Within each unit $\mathcal{U}_u$, the candidate key block and relative RoPE coordinates $\boldsymbol{\ell}_u$ (color bands) are shared, enabling tiled execution with query-specific causal limits. **(Right)** Query rows evaluate attention under causal, full bidirectional, or structurally excluded ($\times$, e.g., excised O) masks, reproducing sequential online representations under exact arithmetic.

where $I_i \subseteq \{1, \dots, N\}$ indexes the candidate key block, $\boldsymbol{\ell}_i \in \mathbb{N}^{|I_i|}$ specifies its logical coordinates, $p_i \in \mathbb{N}$ is the query coordinate, and f_i enforces the causal horizon defining the active visible keys I_i^{vis}. Because state transitions occur strictly at discrete commit boundaries, the candidate key block I_u and its logical coordinates remain invariant throughout each contiguous execution interval $\mathcal{U}_u = \{s_u, \dots, e_u\}$. For $u = u(i)$, we identify $I_i = I_u$ and $\boldsymbol{\ell}_i = \boldsymbol{\ell}_u$:

$$\ell_i(j) \equiv \ell_u(j), \quad \forall i \in \mathcal{U}_u,\ j \in I_u. \tag{7}$$

Equation equation 7 factors query dependence out of key rotations. Consequently, query and key representations within unit $\mathcal{U}_u$ evaluate as standard two-dimensional matrices:

$$\tilde{\mathbf{Q}}_u = \left[\mathbf{R}_{p_i}\mathbf{q}_i\right]_{i\in\mathcal{U}_u} \in \mathbb{R}^{|\mathcal{U}_u|\times d_k}, \qquad \tilde{\mathbf{K}}_u = \left[\mathbf{R}_{\ell_u(j)}\mathbf{k}_j\right]_{j\in I_u} \in \mathbb{R}^{|I_u|\times d_k}, \tag{8}$$

where unrotated vectors $\mathbf{q}_i = \mathbf{W}_Q\bar{\mathbf{h}}_i$ and $\mathbf{k}_j = \mathbf{W}_K\bar{\mathbf{h}}_j$ with $\bar{\mathbf{h}} = \text{RMSNorm}(\mathbf{h})$ are evaluated once per physical event. The attention logit block is then computed via standard two-dimensional matrix multiplication:

$$\mathbf{A}_u = \frac{1}{\sqrt{d_k}}\tilde{\mathbf{Q}}_u\tilde{\mathbf{K}}_u^\top + \mathbf{M}_u^{\text{add}}, \tag{9}$$

where $\mathbf{M}_u^{\text{add}} \in \{0, -\infty\}^{|\mathcal{U}_u|\times|I_u|}$ is an additive logit mask: it is 0 on I_i^{vis} and $-\infty$ outside it $(j > f_i)$. This factorization eliminates the third tensor dimension, restoring two-dimensional Tensor Core execution (Figure 2).

For query $i \in \mathcal{U}_u$, context aggregation evaluates under online softmax normalization:

$$\mathbf{o}_i = \sum_{j\in I_i^{\text{vis}}} \frac{\exp(A_{u,i,j})}{\sum_{j'\in I_i^{\text{vis}}}\exp(A_{u,i,j'})}\mathbf{v}_j, \tag{10}$$

where $\mathbf{v}_j = \mathbf{W}_V\bar{\mathbf{h}}_j$ and $A_{u,i,j}$ denotes the (i,j) entry of logit block $\mathbf{A}_u$. Because each route r_i reconstructs the active key set, causal horizon, and relative logical distances of online state S_t,

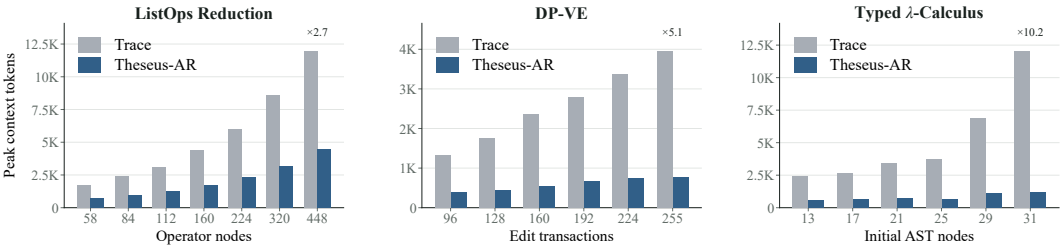


Figure 3: **Peak active context across task complexity.** Median peak active context tokens (W_t) for Transition Trace and Theseus-AR. Dashed line marks the 4K window; annotations report context reduction ratios.

Replay Attention is algebraically equivalent to sequential online execution under exact arithmetic (Appendix B). All replay units execute in parallel via a custom Triton kernel adapting the tiled online-softmax schedule of FlashAttention-2 (Dao, 2024) with unit scheduling, dynamic RoPE synthesis, and shared-key gradient accumulation. Tokenwise projections scale as $\mathcal{O}(N)$, while algorithmic attention complexity scales with active attention pairs $\mathcal{E}_{\text{replay}} = \sum_{i=1}^{N} |I_i^{\text{vis}}| = \Theta(N \cdot \bar{W})$ rather than quadratic history $\Theta(N^2)$ (Proposition B.3; kernel mechanics and operator profiling in Appendix C).

4 Experiments

4.1 Experimental Setup

Tasks. We evaluate three programmatic tasks spanning a *spectrum of state-transition granularities* (Appendix G): (1) **ListOps Reduction (Surgical In-Place Rewrite)**: Localized AST contractions (Nangia & Bowman, 2018) where preserved prefix P and suffix Q remain structurally invariant; (2) **Davis–Putnam Variable Elimination / DP-VE (Dynamic Working-Set Compaction)**: Resolution and unit propagation (Davis & Putnam, 1960; Davis et al., 1962) over an evolving CNF clause multiset; (3) **Typed λ-Calculus (Generative Subterm Duplication)**: Normal-order β-reduction (Church, 1940; 1941; Barendregt, 1984) requiring non-local argument copying across the AST. Figure 3 illustrates how peak active context scales with task complexity across these three regimes.

Protocol, Metrics, and Controls. Our *Stepwise Transition Evaluation* generates candidate transactions from canonical Oracle states under isolated rollout attention and verifies legality via exact domain checkers, isolating transition competence from compounding rollout errors (Appendix E). We decouple evaluation across three axes: (1) *Sample Efficiency (S)*: Cumulative supervised target tokens or source episodes; (2) *Algorithmic Attention Complexity ($\mathcal{E}$)*: Cumulative active query–key attention pairs ($\sum_{i=1}^{N} |I_i^{\text{vis}}|$, Proposition B.3); (3) *Peak Active Context (W_t)*: Median peak active tokens across the transition trajectory, governing both active attention slices during training and active KV-cache length at inference. Using Qwen3-4B (ListOps, DP-VE) and Qwen3-8B (λ-Calculus) full-attention backbones (Appendix A.1), we compare **Transition Trace** against **Theseus-AR** under zero suffix recomputation ($\rho = 0$) and bounded suffix refresh ($\rho = 10$; intermediate $\rho \in \{4, 6, 8\}$ in Appendix A.2). To isolate dynamic RoPE from context-length reduction alone, **Static-Pos** executes identical in-place compaction at $\rho = 10$ while retaining sequential physical tape coordinates.

4.2 Algorithmic Attention Complexity and Learning Dynamics (RQ1)

Sample Acceleration vs. Attention Sparsification. Figure 4 and Table 1 compare optimization dynamics and final validation quality. On **ListOps Reduction**, where surrounding AST coordinates remain invariant, Theseus-AR ($\rho = 10$) reaches the Trace error ceiling in **one quarter** of the supervised budget and reduces terminal action error by $6\times$, an advantage that holds across all operator scales (Appendix Figure 6). On **DP-VE** and **Typed λ-Calculus**, unordered clause sets and subterm copying require full token emission, yielding comparable validation error at matched supervision under the fixed recipe (Figure 4, top). Along the **algorithmic attention complexity axis**, Replay Attention eliminates 54%–86% of active attention pairs at the default $\rho = 10$ frontier; including

Table 1: **Final validation quality and peak active context.** Action error (%), validation NLL, attention pairs, and peak active context tokens (W_t), all read at the validation frontier the table reports. Bold marks the per-task minimum action error and validation NLL. Peak W_t is the median episode peak on the run's own workload: Transition Trace holds the accumulated literal–action stream, Theseus-AR only the largest compacted successor state. Both reductions are measured against the Transition Trace at the same frontier.

Condition	Action Err. ↓	NLL ↓	Episodes	Pairs (B) ↓	Pairs Red. ↓	Peak W_t ↓	Ctx Red.
ListOps							
Transition Trace	1.20	0.00248	4096	19.02	Baseline	3,064	Baseline
Static-Pos ($\rho = 10$)	0.90	0.00157	4096	14.89	-21.7%	1,244	-59.4%
Theseus-AR ($\rho = 0$)	0.70	0.00121	4096	10.61	-44.2%	1,244	-59.4%
Theseus-AR ($\rho = 10$)	**0.20**	**0.00043**	4096	14.89	-21.7%	1,244	-59.4%
DP-VE							
Transition Trace	29.45	**0.09706**	4000	14.52	Baseline	2,700	Baseline
Static-Pos ($\rho = 10$)	31.64	0.11205	4000	6.66	-54.1%	601	-77.7%
Theseus-AR ($\rho = 0$)	34.27	0.13057	4000	4.29	-70.4%	601	-77.7%
Theseus-AR ($\rho = 10$)	**26.89**	0.09841	4000	6.66	-54.1%	601	-77.7%
Typed λ-Calculus							
Transition Trace	1.62	0.00047	4096	156.05	Baseline	6,827	Baseline
Static-Pos ($\rho = 10$)	**1.27**	0.00046	4096	22.17	-85.8%	828	-87.9%
Theseus-AR ($\rho = 0$)	1.50	0.00044	4096	20.48	-86.9%	828	-87.9%
Theseus-AR ($\rho = 10$)	**1.27**	**0.00041**	4096	22.17	-85.8%	828	-87.9%

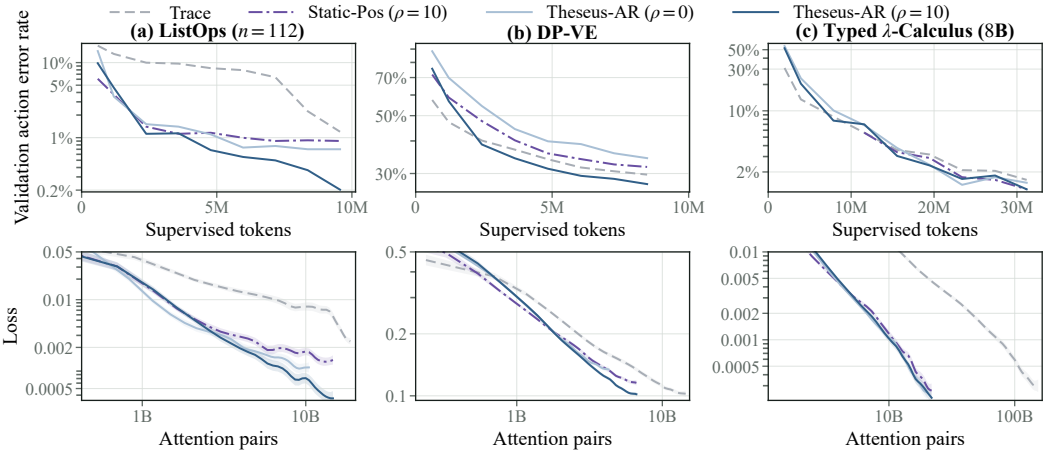


Figure 4: **Learning dynamics across state-transition tasks.** Top row: validation action error against supervised tokens, so the three tasks are compared at matched supervision. Bottom row: training loss against attention pairs, so the same runs are compared at matched algorithmic attention work. Both rows show the same four conditions.

the zero-refresh condition gives 70%–87% (Figure 4, bottom; Table 1; operator latency analysis in Appendix C).

Dynamic RoPE and Suffix Refresh Controls. Compacting context without dynamic RoPE (**Static-Pos**) loses most of the gain on ListOps (4.5× higher error than Theseus-AR) and degrades below uncompacted Trace on DP-VE (Table 1), showing that physical compaction without coordinate realignment induces position drift whenever surviving tokens span wide contexts; on Typed λ-Calculus, where the retained working set is short, Static-Pos matches Theseus-AR and stays ahead of Trace. Comparing $\rho = 0$ and $\rho = 10$ further tests whether skipping suffix re-encoding leaves stale representations: even with zero suffix tokens re-encoded ($\rho = 0$), Theseus-AR already outperforms Trace on ListOps and λ-Calculus, as subsequent queries attend directly to both the inserted replacement V_t and the coordinate-aligned suffix Q_t. On DP-VE, where non-local clause interac-

Table 2: **Stepwise Transition Evaluation against the Transition Trace baseline.** Static-Pos is the static physical-position control at the same refresh horizon, reported as an absolute rate; Δ is Theseus-AR (ρ=10) minus Trace with 95% paired episode-cluster bootstrap CIs (2,000 resamples over paired evaluation episodes, accounting for within-episode step correlation; fixed-checkpoint evaluation uncertainty, not multi-seed training variance); bold gaps exclude zero, arrows mark the improving direction, and Ep. better/worse counts paired episode wins and losses.

Task	Unit	Trace	Static-Pos	Theseus	Δ (95% CI)	Ep. better/worse
ListOps Reduction	Legal err. ↓	3.91	1.77	1.32	$\mathbf{-2.60}$ $\mathbf{[-2.81, -2.38]}$	398/16
ListOps Reduction	Legal success ↑	14.00	37.80	47.00	$\mathbf{+33.00}$ $\mathbf{[+28.80, +37.40]}$	172/7
Typed λ-Calculus	Legal err. ↓	1.64	1.44	1.46	-0.17 $[-0.36, +0.00]$	121/91
Typed λ-Calculus	Legal success ↑	44.50	48.00	48.25	$+3.75$ $[-1.25, +8.50]$	57/42
DP-VE	Legal err. ↓	56.52	59.97	57.10	$\mathbf{+0.57}$ $\mathbf{[+0.14, +1.02]}$	312/380
DP-VE	Legal success ↑	0.00	0.00	0.00	$+0.00$ $[+0.00, +0.00]$	0/0

Task	Phase	Trace	Static-Pos	Theseus	Δ (95% CI)	Ep. better/worse
ListOps Reduction	T1	3.28	1.69	1.99	$\mathbf{-1.29}$ $\mathbf{[-1.58, -1.00]}$	200/53
ListOps Reduction	T2	3.61	1.92	1.09	$\mathbf{-2.51}$ $\mathbf{[-2.82, -2.20]}$	253/15
ListOps Reduction	T3	4.84	1.71	0.87	$\mathbf{-3.97}$ $\mathbf{[-4.34, -3.59]}$	312/5
Typed λ-Calculus	T1	1.75	1.38	1.73	-0.02 $[-0.30, +0.28]$	64/62
Typed λ-Calculus	T2	2.33	2.02	1.78	$\mathbf{-0.55}$ $\mathbf{[-0.90, -0.22]}$	94/64
Typed λ-Calculus	T3	0.86	0.92	0.90	$+0.04$ $[-0.19, +0.27]$	44/45
DP-VE	T1	59.57	64.60	63.13	$\mathbf{+3.56}$ $\mathbf{[+2.89, +4.24]}$	199/423
DP-VE	T2	60.71	66.30	63.62	$\mathbf{+2.91}$ $\mathbf{[+2.17, +3.63]}$	220/373
DP-VE	T3	49.48	49.28	44.86	$\mathbf{-4.62}$ $\mathbf{[-5.40, -3.82]}$	437/191

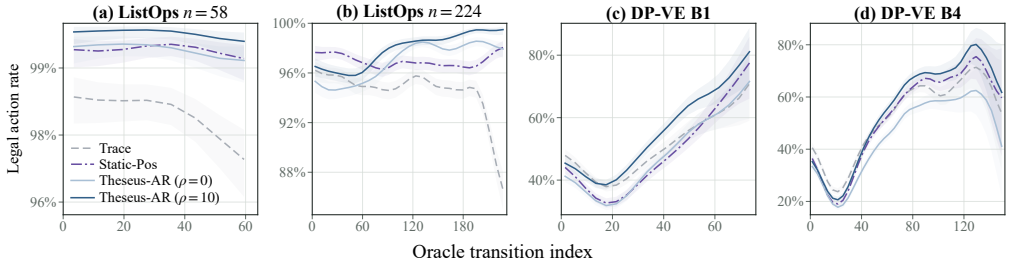


Figure 5: **Stepwise Transition Evaluation across transition horizons.** Legal action rate against Oracle transition index for the first and last ListOps Reduction scales ($n = 58, 224$) and DP-VE buckets ($B1, B4$). Curves compare Transition Trace with Theseus-AR ($\rho = 0$), Theseus-AR ($\rho = 10$), and the Static-Pos control at the same frontier; shaded bands show descriptive pointwise 95% local Wilson intervals over evaluation transitions (formal paired episode-cluster bootstrap CIs accounting for within-episode correlation are reported in Table 2).

tions expose stale conditioning at the edit boundary, refreshing just the first $\rho \leq 10$ suffix tokens resolves the mismatch (saturating by $\rho = 6$; Appendix A.2). Conversely, **State-Recompute** (Appendix A.4) re-encodes the active state at every step (6.3×–27.5× more input tokens per episode). Under a matched processed-input-token frontier, the separately trained State-Recompute condition completes fewer source episodes and remains unconverged; adapter-rank sweeps further show that state topology dominates parameter capacity (Appendix Table 4).

4.3 Deep-Horizon Scaling and Stepwise Transition Validity (RQ2)

Stepwise Validity Across Deep Horizons. Table 2 (reporting paired episode-cluster bootstrap 95% CIs and paired episode win/loss counts) and Figure 5 evaluate stepwise action validity across trajectory tertiles (T_1, T_2, T_3); the complete per-scale and per-bucket curve sets and records are in Appendix D. On **ListOps Reduction**, Trace degrades steadily from T_1 to T_3 as historical rewrites

accumulate, whereas Theseus-AR sustains high stepwise validity into deep horizons ($\Delta = -3.97$ pp in T_3; attention-routing diagnostics in Appendix B), cutting pooled action error by nearly two thirds and more than tripling the zero-error Oracle-episode rate ($14.0\% \rightarrow 47.0\%$). On **Typed λ-Calculus**, Theseus-AR similarly reduces step error during middle-phase term expansion (T_2, $\Delta = -0.55$ pp) and raises the zero-error Oracle-episode rate.

Horizon Phase-Transition in Relational Pruning. On **DP-VE**, pooled action error appears similar between Trace and Theseus-AR, but phase decomposition exposes a sharp horizon crossover (Table 2, bottom; Figure 5c,d). In early and middle phases (T_1, T_2), short transcripts permit direct clause lookup under Trace. In the deep third (T_3), active clause eviction reverses the ordering ($\Delta = -4.62$ pp over Trace) and outperforms Static-Pos.

5 Limitations

Architectural scope. Theseus-AR relies on individually addressable key–value entries to excise intermediate spans and rotate suffix coordinates with zero residual interference. Accordingly, our evaluation uses pure full-attention backbones (Qwen3-4B/8B) rather than hybrid architectures that interleave state-space models (SSMs) or linear-attention layers, where historical tokens are additively entangled inside a recurrent state matrix.

Supervision and evaluation scope. Our experiments focus on programmatic tasks with deterministic verifiers and explicit transition trajectories; because Stepwise Transition Evaluation anchors each step at canonical Oracle states to isolate single-step transition validity, scaling to open-ended natural language and repository-level software engineering requires broader trajectory acquisition and end-to-end closed-loop evaluation. In addition, primary training comparisons share a single fixed random seed per task (paired across scales, adapter capacities, and refresh horizons), so reported confidence intervals quantify evaluation-sample uncertainty for fixed checkpoints rather than multi-seed training variance.

Kernel wall-clock latency. While Replay Attention reduces algorithmic attention complexity from $\Theta(N^2)$ to $\Theta(N \cdot \bar{W})$, our Triton implementation remains a prototype and does not yet translate this reduction into net wall-clock speedups over mature dense attention kernels (Appendix C). We report the algorithmic reduction as our result and leave operator-level optimization for future work.

6 Conclusion

This work moves autoregressive generation beyond monotonic prefix extension toward selective discrete recurrence. By pairing Theseus-AR—which evolves an explicit sequence state in place through dynamic RoPE coordinate alignment and key–value compaction—with Replay Attention—which compiles time-varying state views into trajectory-parallel training with exact-arithmetic online equivalence—pretrained Transformers can iteratively revise intermediate structures without re-encoding successor states from scratch. Controlled evaluations across programmatic tasks show that retiring superseded states reduces cumulative visible attention pairs from $\Theta(N^2)$ to $\Theta(N \cdot \bar{W})$ and bounds active inference context by the current state workload ($\Theta(W_t)$), accelerating convergence when surrounding structure is invariant and improving deep-horizon transition validity under Stepwise Transition Evaluation.

More broadly, formulating generation as iterative transitions over a *structured context* shifts the role of the context window from a passive, append-only transcript to an actively maintained working state. This paradigm opens two complementary directions for scaling beyond programmatic testbeds. First, in domains such as software engineering and interactive authoring, models can learn directly from version-control commit histories and editor telemetry, or from synthetic trajectories distilled out of linear reasoning chains. Second, discrete in-place updates provide a natural action space for reinforcement learning: rather than policies that only append tokens until context exhaustion, agents can learn when to scaffold intermediate subgoals and when to collapse completed reasoning steps under environment or verifier reward. Coupled with kernel-level optimizations and future closed-loop rollout training, selective discrete recurrence offers a path toward bounded-context, long-horizon autoregressive generation.

AI USE STATEMENT

OpenAI Codex was used as an AI coding and research assistant during the implementation and preparation of this work. Its use included drafting and refactoring code, debugging, experiment orchestration, analysis and plotting scripts, table generation, and language editing. Codex was not treated as an independent source of scientific evidence: the authors made and approved the scientific decisions, inspected generated code and outputs, verified the mathematical derivations, citations, quantitative results, and claims, and take full responsibility for the final manuscript and released artifacts.

REPRODUCIBILITY STATEMENT

To preserve anonymous peer review, the implementation and accompanying reproducibility materials are not publicly available during review. Upon acceptance, we will publicly release the source code, experiment configurations, evaluation and analysis scripts, figure and table generators, and accompanying documentation. The method and training procedure are described in Section 3. The experimental setup, backbone isolation criteria, optimization settings, and compute hardware are given in Appendix A.1; the programmatic testbeds and data-generation procedures are documented in Appendix G; and the processed-input-token accounting used as the cost axis is defined in Appendix C.1. The Stepwise Transition Evaluation protocol is formalized in Appendix E, and its metric contracts and complete per-task records are provided in Appendix D. Appendix B derives the Replay Attention routing model, its active-attention-pair complexity bound, and its algebraic equivalence to online execution under exact arithmetic; Appendix C qualifies the current operator numerically and reports its execution-cost boundary; Appendix F specifies the Theseus-AR transition semantics and cache lifecycle; and Appendix H prints representative materialized episodes of the three tasks.

The post-acceptance release will include the scripts and configurations needed to regenerate the reported figures and tables, rebuild the testbeds and their Oracle trajectories from the released generators, and rerun the Stepwise Transition Evaluation protocol. Hardware and software requirements will be documented with the corresponding configurations. Results depending on external model weights or datasets remain subject to their original licenses, which will be identified in the released documentation.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021. URL https://papers.nips.cc/paper_files/paper/2021/hash/958c530554f78bcd8e97125b70e6973d-Abstract.html.

Sanket Badhe, Priyanka Tiwari, and Jonghyun Chung. SKILL.state: Scalable long-horizon agent skills. *arXiv preprint arXiv:2608.26263*, 2026. doi: 10.48550/arXiv.2608.26263. URL https://arxiv.org/abs/2608.26263.

Henk P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, revised edition, 1984. ISBN 0-444-87508-5.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022. doi: 10.48550/arXiv.2207.14255. URL https://arxiv.org/abs/2207.14255.

Alonzo Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5(2): 56–68, 1940. doi: 10.2307/2266170. URL https://doi.org/10.2307/2266170.

Alonzo Church. *The Calculi of Lambda-Conversion*, volume 6 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, 1941.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960. doi: 10.1145/321033.321034. URL https://doi.org/10.1145/321033.321034.

Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962. doi: 10.1145/368273.368557. URL https://doi.org/10.1145/368273.368557.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, volume 32, pp. 11181–11191. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/675f9820626f5bc0afb47b57890b466e-Abstract.html.

Kairong Han, Nuanqiao Shan, Ziyu Zhao, Zijing Hu, Xinpeng Dong, Junjian Ye, Lujia Pan, Fei Wu, and Kun Kuang. C^2DLM: Causal concept-guided diffusion large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 821–838, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.40. URL https://aclanthology.org/2026.findings-acl.40/.

Zhenyu He, Jun Zhang, Shengjie Luo, Jingjing Xu, Zhi Zhang, and Di He. Let the code LLM edit itself when you edit the code. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2407.03157. URL https://arxiv.org/abs/2407.03157.

Vasilis Kontonis, Yuchen Zeng, Shivam Garg, Lingjiao Chen, Hao Tang, Ziyan Wang, Ahmed Awadallah, Eric Horvitz, John Langford, and Dimitris Papailiopoulos. MEMENTO: Teaching LLMs to manage their own context. *arXiv preprint arXiv:2604.09852*, 2026. doi: 10.48550/arXiv.2604.09852. URL https://arxiv.org/abs/2604.09852.

Nikita Nangia and Samuel R. Bowman. ListOps: A diagnostic dataset for latent tree learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 92–99. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-4013. URL https://doi.org/10.18653/v1/N18-4013.

Gleb Rodionov, Roman Garipov, Alina Shutova, George Yakushev, Erik Schultheis, Vage Egiazarian, Anton Sinitsin, Denis Kuznedelev, and Dan Alistarh. Hogwild! inference: Parallel LLM generation via concurrent attention. In *Advances in Neural Information Processing Systems*, volume 38, pp. 52014–52055. Curran Associates, Inc., 2025. doi: 10.52202/085713-1551. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4293727fc48c60b4adfe7a8e44f6a446-Abstract-Conference.html.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130136–130184. Curran Associates, Inc., 2024. doi: 10.52202/079017-4135. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb0b13cc515724ab8015bc978fdde0ad-Abstract-Conference.html.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008. Curran Associates, Inc., 2017. URL https://papers.nips.cc/paper/7181-attention-is-all-you-need.

Buqiang Xu, Zirui Xue, Dianmou Chen, Chenyang Fu, Chiyu Wu, Caiying Huang, Chen Jiang, Jizhan Fang, Xinle Deng, Yijun Chen, Yunzhi Yao, Xuehai Wang, Jin Shang, Gong Yu, and Ningyu Zhang. TokenPilot: Cache-efficient context management for LLM agents. *arXiv preprint arXiv:2606.17016*, 2026. doi: 10.48550/arXiv.2606.17016. URL https://arxiv.org/abs/2606.17016.

Shuchen Xue, Tianyu Xie, Tianyang Hu, Zijin Feng, Jiacheng Sun, Kenji Kawaguchi, Zhenguo Li, and Zhi-Ming Ma. Any-order GPT as masked diffusion model: Decoupling formulation and architecture. *arXiv preprint arXiv:2506.19935*, 2025. doi: 10.48550/arXiv.2506.19935. URL https://arxiv.org/abs/2506.19935.

Chenxiao Yang, Nathan Srebro, David McAllester, and Zhiyuan Li. PENCIL: Long thoughts with short memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 71092–71156. PMLR, 2025. URL https://proceedings.mlr.press/v267/yang25ac.html.

Chenxiao Yang, Nathan Srebro, and Zhiyuan Li. Recursive models for long-horizon reasoning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=nERMuZtneC.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, pp. 20359–20405. Curran Associates, Inc., 2025. doi: 10.52202/085713-0605. URL https://doi.org/10.52202/085713-0605.

A Ablation Studies and Extended Empirical Analysis

A.1 Experimental Setup

Primary runs use Qwen3-4B (ListOps Reduction, DP-VE) and Qwen3-8B (Typed λ-Calculus) decoder-only Transformers with full attention in every layer (Grouped-Query Attention, RMSNorm, SwiGLU, and RoPE without learned absolute position biases). Unlike sliding-window or hybrid recurrent/state-space architectures, full-attention layers store the complete causal state in the key–value cache, so masking excised physical keys strictly retracts their contribution without state contamination. Unless otherwise ablated, all runs attach LoRA adapters of rank $r = 32$ on W_v, W_o, and the three MLP projections and rank $r = 64$ on W_q and W_k ($\alpha/r = 2$, dropout 0.05), optimized with AdamW (initial learning rate 1.0×10^{-4}, weight decay 0, gradient clipping 1.0, cosine decay to 10% over $1{,}024$ updates without warmup) in bfloat16 using FlashAttention-2 for serialized baselines and Triton Replay Attention for Theseus-AR. Qwen3-4B runs execute on an RTX 5090 D V2 and Qwen3-8B runs on an NVIDIA A800. All conditions within a task share a fixed training seed (43) and identical episode presentation order for a paired comparison; rather than repeating single configurations across multiple training seeds, we examine whether relative condition orderings remain consistent across four operator scales ($n \in \{29, 58, 84, 112\}$), five adapter rank capacities (39.0M–120.4M parameters), and five refresh strides ($\rho \in \{0, 4, 6, 8, 10\}$). Throughout the paper, reported intervals quantify evaluation-sample uncertainty for fixed checkpoints rather than run-to-run variance across training seeds.

A.2 Boundary Refresh Spectrum and Capacity Ablations

Boundary Refresh Horizon Spectrum ($\rho \in \{0, 4, 6, 8, 10\}$). Table 3 reports the complete validation matrix across $\rho \in \{0, 4, 6, 8, 10\}$, showing how transition granularity governs suffix recontextualization. On localized AST contractions (ListOps), zero suffix recomputation ($\rho = 0$) already outperforms Transition Trace, and validation error decreases monotonically through $\rho = 10$. On relational clause sets (DP-VE), where newly asserted resolvents interact non-locally with surviving clauses, unrefreshed suffixes ($\rho = 0$) leave boundary staleness that is resolved once a small frontier of $\rho = 6$ suffix tokens is re-encoded. Beyond $\rho = 6$, validation accuracy saturates on both DP-VE and Typed λ-Calculus while expanding ρ from 6 to 10 increases processed input tokens and visible attention pairs on DP-VE by approximately 15%, indicating that a narrow $\mathcal{O}(1)$ refresh window suffices.

Adapter Rank Capacity Control. Table 4 sweeps five matched MLP/QK LoRA rank settings ($16/16$ to $64/64$, spanning 39.0M to 120.4M trainable parameters) on ListOps Reduction ($n = 112$). Across every setting, Theseus-AR ($\rho = 10$) improves monotonically with adapter capacity and remains $4.6\times$ to $13.1\times$ lower in action error than the matched-rank Transition Trace run; even the smallest Theseus-AR adapter (39.0M) achieves less than half the action error of the widest Trace adapter (120.4M), whereas Trace plateaus above $32/32$ and degrades sharply when MLP rank drops to 16.

A.3 Multi-Scale Scaling on ListOps Reduction

Figure 6 and Table 7 evaluate optimization trajectories across four operator budgets ($n \in \{29, 58, 84, 112\}$). At every scale, Theseus-AR ($\rho = 10$) reaches Transition Trace's terminal action error within 25%–50% of its own supervised-token budget and converges to a substantially lower final error. In every panel, Static-Pos falls between Transition Trace and logical-position Theseus-AR, with the penalty of unaligned physical coordinates widening monotonically as operator depth n grows.

A.4 State-Recompute Cost Under a Processed-Token Frontier

Unlike Theseus-AR, **State-Recompute** discards historical key–value representations and freshly re-encodes the entire active state X_t at every transition. Because State-Recompute conditions each decision on a standalone state without prior trace, evaluating a Transition Trace or Theseus-AR checkpoint on isolated intermediate states would confound representation mode with a train–test

Table 3: **Complete final validation condition matrix.** Entries are action error and token error (%), validation NLL, processed input tokens in millions, cumulative causal-visible attention pairs in billions, and the completed training frontier in source episodes; every metric and the pair total are read at the frontier printed with the row, and bold marks the minimum action error, token error, and validation NLL inside the panel the row is printed under, so the two DP-VE budgets are bolded independently. The backbone assignments are given in the experimental setup, and Typed λ-Calculus lists only the refresh strides that were trained ($\rho \in \{0, 6, 10\}$). The DP-VE panel is split by budget: the $4{,}000$-episode rows re-evaluate the exported checkpoint on the shared validation partition, and the $8{,}000$-episode continuation is listed separately for the four conditions that were continued. The pair totals of the refresh strides are derived from the no-refresh realization of the same series through the exact suffix-refresh identity, verified per update against each stride's own query-token counts.

Condition	Action Err. ↓	Token Err. ↓	Val. NLL ↓	Input Tokens (M)	Attn. Pairs (B)	Episodes
ListOps						
Transition Trace	1.20	0.06	0.00248	12.46	19.02	4096
Static-Pos ($\rho = 10$)	0.90	0.05	0.00157	16.80	14.89	4096
Theseus-AR ($\rho = 0$)	0.70	0.04	0.00121	12.46	10.61	4096
Theseus-AR ($\rho = 4$)	0.48	0.03	0.00085	14.27	12.38	4096
Theseus-AR ($\rho = 6$)	0.46	0.02	0.00080	15.13	13.23	4096
Theseus-AR ($\rho = 8$)	0.33	0.02	0.00054	15.98	14.07	4096
Theseus-AR ($\rho = 10$)	**0.20**	**0.01**	**0.00043**	16.80	14.89	4096
DP-VE ($4{,}000$ episodes)						
Transition Trace	29.45	**3.55**	**0.09706**	10.77	14.52	4000
Static-Pos ($\rho = 10$)	31.64	4.07	0.11205	16.64	6.66	4000
Theseus-AR ($\rho = 0$)	34.27	4.64	0.13057	10.77	4.29	4000
Theseus-AR ($\rho = 4$)	27.56	3.78	0.10408	13.38	5.35	4000
Theseus-AR ($\rho = 6$)	**26.61**	3.62	0.10022	14.49	5.80	4000
Theseus-AR ($\rho = 8$)	27.24	3.67	0.10017	15.57	6.23	4000
Theseus-AR ($\rho = 10$)	26.89	3.57	0.09841	16.64	6.66	4000
DP-VE ($8{,}000$-episode continuation)						
Transition Trace	27.12	**3.18**	0.08694	21.54	29.06	8000
Theseus-AR ($\rho = 0$)	31.53	4.00	0.11294	21.54	8.59	8000
Theseus-AR ($\rho = 6$)	25.61	3.26	0.08834	28.99	11.60	8000
Theseus-AR ($\rho = 10$)	**25.24**	3.24	**0.08630**	33.30	13.33	8000
Typed λ-Calculus						
Transition Trace	1.62	0.02	0.00047	32.90	156.05	4096
Static-Pos ($\rho = 10$)	**1.27**	**0.01**	0.00046	26.65	22.17	4096
Theseus-AR ($\rho = 0$)	1.50	0.02	0.00044	32.90	20.48	4096
Theseus-AR ($\rho = 6$)	**1.27**	**0.01**	0.00044	34.67	21.57	4096
Theseus-AR ($\rho = 10$)	**1.27**	**0.01**	**0.00041**	35.64	22.17	4096

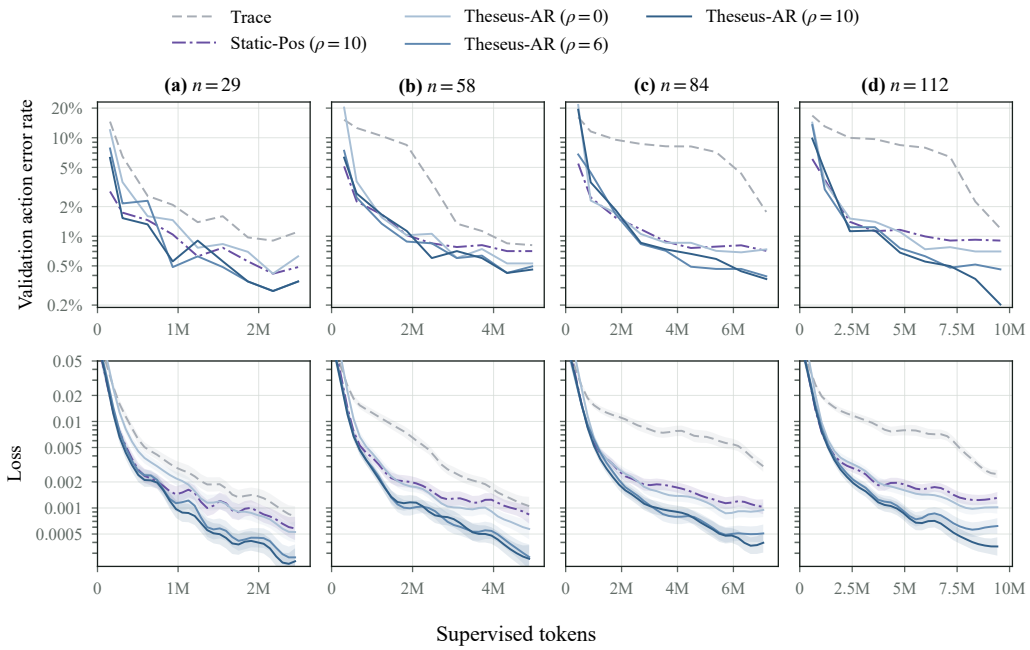


Figure 6: **ListOps Reduction learning dynamics across operator budgets.** Validation action error (top) and smoothed training loss (bottom) against cumulative supervised tokens across four operator budgets ($n \in \{29, 58, 84, 112\}$); shaded bands show descriptive 95% local evaluation-set intervals at each checkpoint. Each panel draws Transition Trace, the **Static-Pos** control (identical in-place compaction with physical tape coordinates at $\rho = 10$), and the logical-position Theseus-AR family ($\rho \in \{0, 6, 10\}$). Theseus-AR ($\rho = 10$) reaches Trace's terminal action error after approximately one quarter to one half of its own final supervised-token total, depending on the operator budget; the Static-Pos curves show the contribution of compaction without logical-position alignment.

Table 4: **LoRA capacity control on ListOps Reduction.** All runs use Qwen3-4B, the same data order, optimizer, scheduler, seed, and 4,096 training episodes; only the adapter topology changes. Ranks are listed as MLP/QK; v/o stays at rank 32 and every adapter keeps $\alpha/r = 2$ with dropout 0.05. Trainable parameters count the LoRA adapters and the eleven trainable control-token embedding rows; the 48-episode validation grid resolves one action to 0.018 percentage points. Bold marks the best value in each column.

Condition	MLP/QK rank	Train. params	Action Err. (%) ↓	Token Err. (%) ↓	Val. NLL ↓
Transition Trace	16/16	39.0M	5.0701	0.2492	0.007186
Transition Trace	32/16	60.2M	4.5723	0.2242	0.006368
Transition Trace	32/32	66.1M	1.1984	0.0607	0.002395
Transition Trace	32/64	77.9M	1.1984	0.0598	0.002480
Transition Trace	64/64	120.4M	1.2721	0.0643	0.002451
Theseus-AR ($\rho = 10$)	16/16	39.0M	0.4978	0.0259	0.000708
Theseus-AR ($\rho = 10$)	32/16	60.2M	0.3503	0.0170	0.000465
Theseus-AR ($\rho = 10$)	32/32	66.1M	0.2581	0.0125	0.000513
Theseus-AR ($\rho = 10$)	32/64	77.9M	0.2028	0.0107	0.000425
Theseus-AR ($\rho = 10$)	64/64	120.4M	**0.1659**	**0.0089**	**0.000306**

framing mismatch; all State-Recompute records in Figure 7 and Table 5 are therefore trained natively under the state-only contract. Re-encoding every intermediate state incurs $\Theta(\sum_{t=0}^{T-1} |X_t|)$ input tokens per episode (6.3×–27.5× more processed input tokens per episode than Theseus-AR at $\rho = 10$, alongside an additional unmeasured $\Theta(\sum_{t=0}^{T-1} |X_t|^2)$ dense-attention cost). Under a matched processed-input-token frontier, the separately trained State-Recompute condition completes fewer

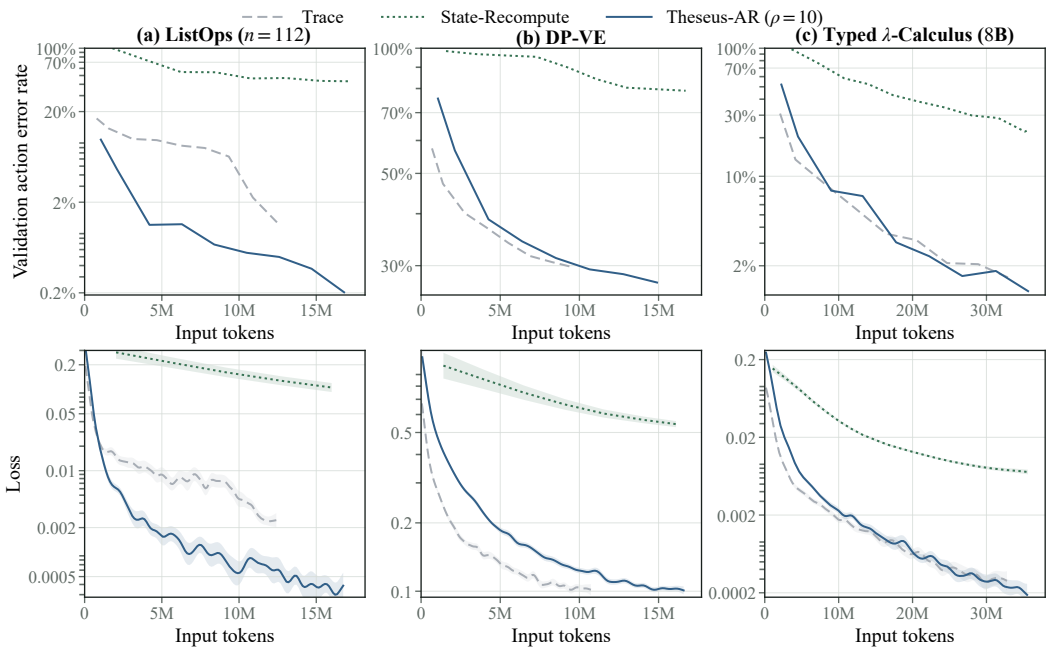


Figure 7: **State-Recompute cost under a processed-token frontier.** Validation action error (top) and training loss (bottom) against cumulative processed input tokens—the tokens each run had to pass through the encoder—for Transition Trace, State-Recompute, and Theseus-AR ($\rho = 10$). State-Recompute re-encodes the whole active state at every transition, so on this axis it buys far fewer source episodes per processed token wherever the trajectory is long enough to make re-encoding dominant, and it stays well above the two lineage-preserving realizations in every panel. Each State-Recompute curve ends at its own stop budget; on DP-VE that budget leaves it at 79.03% action error after 16.71M processed tokens, past the 14.90M of the Theseus-AR ($\rho = 10$) curve and the 9.64M of the Transition Trace curve, both of which last validate at $3{,}584$ episodes.

Table 5: **State-Recompute cost under a processed-token slice.** State-Recompute re-encodes the active state at every transition, so it cannot be carried to the episode frontiers of Table 3 within a comparable compute envelope. Each State-Recompute row was stopped at the budget printed with it, and each budget was chosen so that the run's processed input tokens land at approximately those of the Theseus-AR ($\rho = 10$) run it is paired with. *Input k/ep* is processed input tokens per completed source episode, the quantity that makes those frontiers prohibitive; against the paired run it is ListOps Reduction $10.5\times$–$27.5\times$, DP-VE $19.0\times$, Typed λ-Calculus $6.3\times$. Episodes, supervised tokens and the teacher-forced action/token error and NLL are read at each run's own last validation record, so the metric columns are end-of-slice snapshots rather than a converged quality comparison; the $4{,}096$-episode rows report the last validation their own schedule reached.

Task	Condition	Setting	Stop budget	Episodes	Input M	Input k/ep	Sup. M	Action Err. (%)	Token Err. (%)	Val. NLL
ListOps Reduction	State-Recompute	n=29	152 episodes	152	1.85	12.20	0.09	31.53	1.97	0.056097
ListOps Reduction	Theseus-AR ($\rho = 10$)	n=29	4,096-episode schedule	4096	4.74	1.16	2.49	0.35	0.02	0.000492
ListOps Reduction	State-Recompute	n=58	152 episodes	152	5.48	36.08	0.18	37.50	2.27	0.066947
ListOps Reduction	Theseus-AR ($\rho = 10$)	n=58	4,096-episode schedule	4096	8.94	2.18	4.95	0.46	0.02	0.001018
ListOps Reduction	State-Recompute	n=84	152 episodes	152	10.30	67.76	0.27	38.43	2.29	0.062420
ListOps Reduction	Theseus-AR ($\rho = 10$)	n=84	4,096-episode schedule	4096	12.72	3.11	7.18	0.37	0.02	0.000570
ListOps Reduction	State-Recompute	n=112	152 episodes	152	17.13	112.71	0.35	43.18	2.69	0.071096
ListOps Reduction	Theseus-AR ($\rho = 10$)	n=112	4,096-episode schedule	4096	16.80	4.10	9.56	0.20	0.01	0.000425
DP-VE	State-Recompute	$\rho = 10$	212 episodes	212	16.71	78.82	0.50	79.03	17.47	0.490475
DP-VE	Theseus-AR ($\rho = 10$)	$\rho = 10$	4,096-episode schedule	3584	14.90	4.16	8.48	27.32	3.73	0.101170
Typed λ-Calculus	State-Recompute	$\rho = 10$	163 steps	652	35.80	54.91	5.02	23.16	0.27	0.007306
Typed λ-Calculus	Theseus-AR ($\rho = 10$)	$\rho = 10$	4,096-episode schedule	4096	35.64	8.70	31.17	1.27	0.01	0.000413

source episodes and remains unconverged. Table 5 therefore serves as a workload-cost measurement rather than a converged quality comparison.

Table 6: **Matched-frontier validation records for the Static-Pos baseline.** Entries are final validation action error and token error (%), validation NLL, and completed training episodes. Static-Pos keeps the naive physical tape coordinates of the compacted state with $\rho = 10$; the logical-position Theseus-AR comparison uses the same refresh horizon.

Task	Condition	Action Err. ↓	Token Err. ↓	Val. NLL ↓	Episodes
ListOps	Transition Trace	1.20	0.06	0.00248	4096
ListOps	Static-Pos ($\rho = 10$)	0.90	0.05	0.00157	4096
ListOps	Theseus-AR ($\rho = 10$)	0.20	0.01	0.00043	4096
DP-VE	Transition Trace	29.45	3.55	0.09706	4000
DP-VE	Static-Pos ($\rho = 10$)	31.64	4.07	0.11205	4000
DP-VE	Theseus-AR ($\rho = 10$)	26.89	3.57	0.09841	4000
Typed λ-Calculus	Transition Trace	1.62	0.02	0.00047	4096
Typed λ-Calculus	Static-Pos ($\rho = 10$)	1.27	0.01	0.00046	4096
Typed λ-Calculus	Theseus-AR ($\rho = 10$)	1.27	0.01	0.00041	4096

Table 7: **Static-Pos across operator budgets on ListOps Reduction.** Final validation action and token error (%), validation NLL, and completed training episodes at the shared 4k frontier. Each budget is its own release and its own $4{,}096$-episode / $1{,}024$-update run, so the three conditions inside a block differ only in the state-tracking condition; Static-Pos keeps the physical tape coordinates of the compacted state at the same refresh horizon as the logical-position Theseus-AR row. The 48 validation episodes resolve one action to $0.069/0.035/0.025/0.018$ percentage points at $n = 29/58/84/112$.

Condition	Nodes	Action Err. ↓	Token Err. ↓	Val. NLL ↓	Episodes
Transition Trace	29	1.11	0.05	0.00198	4096
Transition Trace	58	0.81	0.04	0.00229	4096
Transition Trace	84	1.76	0.09	0.00369	4096
Transition Trace	112	1.20	0.06	0.00248	4096
Static-Pos ($\rho = 10$)	29	0.49	0.02	0.00072	4096
Static-Pos ($\rho = 10$)	58	0.71	0.03	0.00196	4096
Static-Pos ($\rho = 10$)	84	0.71	0.04	0.00150	4096
Static-Pos ($\rho = 10$)	112	0.90	0.05	0.00157	4096
Theseus-AR ($\rho = 10$)	29	0.35	0.02	0.00049	4096
Theseus-AR ($\rho = 10$)	58	0.46	0.02	0.00102	4096
Theseus-AR ($\rho = 10$)	84	0.37	0.02	0.00057	4096
Theseus-AR ($\rho = 10$)	112	0.20	0.01	0.00043	4096

A.5 Extended Static-Pos Baseline Records

Tables 6 and 7 report the full validation metrics for **Static-Pos**, which performs identical in-place compaction at $\rho = 10$ while leaving surviving suffix keys at their static physical tape indices instead of applying dynamic RoPE shifts ($\Delta_t = |V_t| - |O_t|$). Across both tables, the cost of omitting coordinate realignment scales directly with the span of the retained state: when the compacted working set stays short (Typed λ-Calculus and ListOps at $n = 29$), physical coordinates remain nearly sufficient and the gap to Theseus-AR registers mainly in validation NLL; as the active tape lengthens across ListOps budgets ($n = 29 \rightarrow 112$), the action-error penalty of Static-Pos grows monotonically (from 2 to 38 validation actions), and on unordered DP-VE clause multisets without bracket hierarchy, unadjusted coordinates degrade performance below uncompacted Transition Trace.

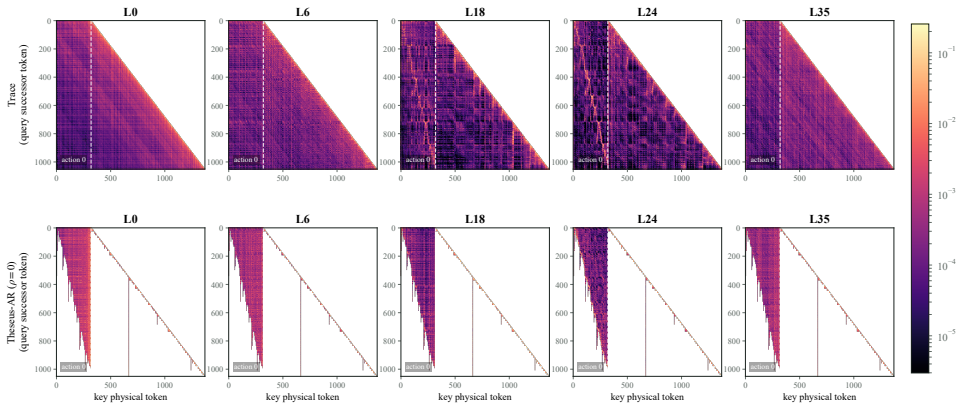


Figure 8: **Causal attention routing across layer depths ($L_0, L_6, L_{18}, L_{24}, L_{35}$).** Evaluated on a fixed 64-reduction ListOps episode (leading prompt excluded from both axes). Transition Trace (top row) disperses conditional attention across accumulated historical rewrites as depth increases, whereas Theseus-AR (bottom row) restricts attention mass to the active diagonal ribbon.

B Replay Attention: Mechanistic Routing, Operator Formulation, and Algorithmic Equivalence

This appendix formalizes the attention routing diagnostics, algorithmic complexity reduction, replay unit compilation model, and algebraic equivalence proof for Replay Attention (Section 3). We first analyze empirical attention routing across network depths and reduction steps on a frozen ListOps analysis episode, then prove the $\Theta(N \cdot \bar{W})$ algorithmic attention complexity bound (Proposition B.3), and finally establish algebraic equivalence between compiled Replay Attention and sequential online execution under exact arithmetic (Proposition B.5).

B.1 Mechanistic Attention Sparsification Visualized

Standard autoregressive training over a flattened historical trace exposes every preceding physical event to causal queries, incurring quadratic algorithmic attention complexity ($\Theta(N^2)$ active attention pairs). Replay Attention instead compiles each query's active visibility view I_i^{vis} and excludes retired or superseded entries before softmax normalization.

Physical Tape Sparsification Across Network Depths. Figure 8 visualizes row-conditional causal attention matrices between physical query and key events across five representative network depths ($L_0, L_6, L_{18}, L_{24}, L_{35}$).

- **Transition Trace (Dense Lower Triangle)**: In early layers (L_0, L_6), attention exhibits localized diagonal scanning over immediate syntactic neighbors. As network depth increases ($L_{18} \to L_{35}$), query attention disperses broadly across the lower-triangular historical transcript, allocating substantial conditional mass to already-excised intermediate steps.
- **Theseus-AR (Sparse Active Ribbon)**: Under Replay Attention, inactive physical keys corresponding to superseded subtrees are masked out, confining conditional attention mass to a narrow ribbon tracking the active sequence state boundary.

Logical AST Node Frontier Tracking Across Horizons. While Figure 8 visualizes physical tape indices, Figure 9 aggregates query-to-key mass onto logical AST nodes across successive Oracle reduction steps. As subtrees contract, Theseus-AR concentrates conditional attention mass along a diagonal trail tracking the leftmost ready operator node across layers, whereas Transition Trace leaks conditional mass horizontally across previously evaluated subtrees in the historical transcript.

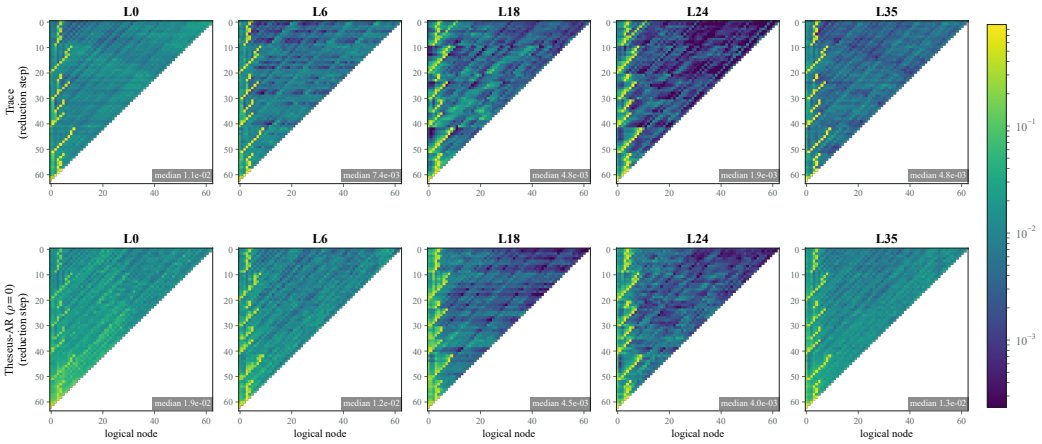


Figure 9: **Logical AST node attention allocation across Oracle reduction steps.** Conditional attention mass aggregated onto logical AST nodes across five layers ($L_0, L_6, L_{18}, L_{24}, L_{35}$) on the same 64-reduction episode. Rows correspond to sequential reduction steps; columns correspond to logical AST nodes ordered from left to right. Theseus-AR maintains a concentrated diagonal trail tracking the active reduction frontier, whereas Transition Trace disperses mass horizontally across historical subtrees.

Layer	Region entropy ↓			Node top-1 ↑			Region locality ↑		
	Trace	Theseus-AR	Δ	Trace	Theseus-AR	Δ	Trace	Theseus-AR	Δ
L_0	0.734	0.899	0.165	0.590	0.133	-0.458	0.532	0.886	0.353
L_{18}	0.630	0.849	0.219	0.544	0.374	-0.171	0.659	0.809	0.150
L_{35}	0.630	0.816	0.186	0.780	0.369	-0.410	0.651	0.918	0.267

Table 8: Quantitative attention summary for one fixed ListOps Reduction episode (112 operator nodes, balanced shape) under the paired Qwen3-4B Trace and Theseus-AR ($\rho = 0$) checkpoints. The leading prompt and initialization action are excluded from both axes, and each scope is renormalized per (query row, head) before measuring its statistics. Each entry is the median across query rows of the head-averaged metric (the "head-mean median"). Region entropy is normalized by $\log K$; locality is conditional mass within the configured ± 32 physical-token window, not total attention mass. Δ is Theseus-AR $-$ Trace: positive entropy differences indicate a more diffuse conditional region distribution, whereas positive top-1/locality differences indicate stronger concentration or local routing.

B.2 Quantitative Attention Diagnostics: Conditional Locality and Concentration

Table 8 and Figures 10–11 quantify attention routing on one fixed ListOps Reduction episode (112 operator nodes, balanced shape), evaluated with the paired Qwen3-4B Trace and Theseus-AR ($\rho = 0$) checkpoints. Excluding the leading prompt and initialization action from both query and key axes, each (query row, head) distribution is renormalized over the retained successor region to measure conditional region locality (± 32 physical token window), conditional region entropy, and logical-node top-1 share.

Dissecting the Locality vs. Single-Node Concentration Mechanism. The quantitative diagnostics highlight two complementary routing behaviors:

1. **Conditional region locality**: Across layers, the Theseus-AR ($\rho = 0$) condition has a mean per-layer median of 0.915 within the retained successor region, compared with 0.613 for Trace. At L_{35} the corresponding head-mean medians are 0.918 and 0.651, confirming that Replay Attention keeps conditional routing tightly focused within the local active window.

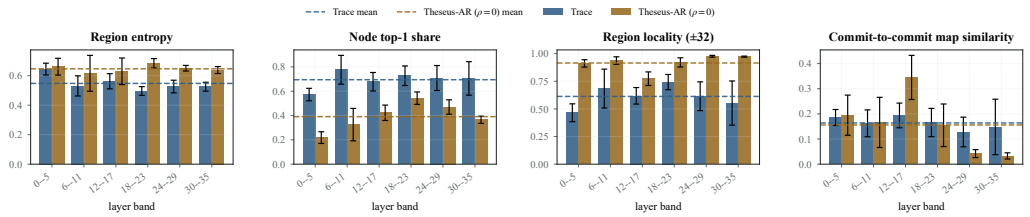


Figure 10: **Layer-wise attention routing statistics across contiguous six-layer bands.** Bars report the mean of per-layer medians within each band; whiskers show standard deviations across layers. Dashed lines show condition-wide means over all 36 layers. Theseus-AR sustains higher conditional region locality (0.915 overall mean vs. 0.613 for Trace) while maintaining moderate node top-1 share, reflecting multi-constituent integration within the active workspace.

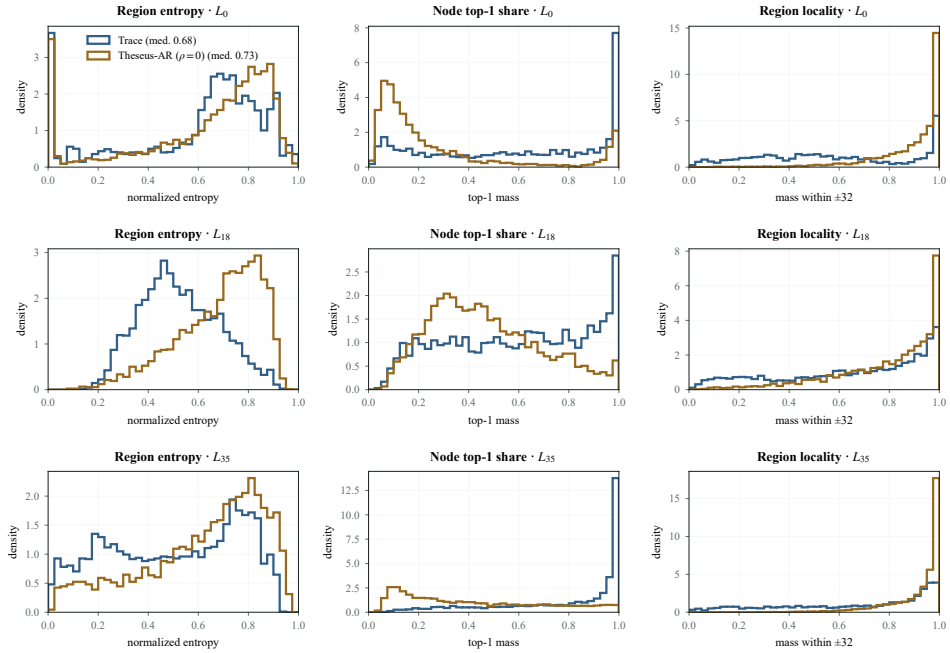


Figure 11: **Distributional attention diagnostics at early, middle, and final layers (L_0, L_{18}, L_{35}).** Density histograms over query–head pairs for region entropy, logical-node top-1 share, and region locality. At L_{35}, Transition Trace displays a pronounced spike near node top-1 $= 1.0$ coupled with lower conditional region locality, whereas Theseus-AR concentrates mass near locality $= 1.0$ while distributing weights across interacting elements of the active redex.

2. **Conditional concentration versus locality**: At L_{35}, Trace has a higher conditional node top-1 share (0.780 vs. 0.369) alongside lower conditional locality (0.651 vs. 0.918), indicating that substantial post-prompt attention mass concentrates onto single distant historical nodes. In contrast, evaluating an AST redex (e.g., contracting `[max 4 [min 2 5] 3]` to `[max 4 2 3]`) fundamentally requires attending simultaneously to multiple constituent tokens—the operator and its sibling operands. Theseus-AR has higher conditional region entropy at L_{35} (0.816 vs. 0.630), reflecting compositional integration across the operator and its sibling operands within the local workspace.

Commit-to-Commit Routing Dynamics. Commit-to-commit map cosine similarity remains comparable between conditions (0.156 for Theseus-AR vs. 0.164 for Trace when averaged over layers), with the modest late-layer reduction under Theseus-AR reflecting route updates as dynamic RoPE shifts successor coordinates.

B.3 Proposition 2: Algorithmic Attention Complexity

Proposition 2 (Algorithmic Attention Pair Complexity). *Let τ_T^* be a state-transition trajectory with total physical event count N, partitioned into M replay units $\mathcal{U}_1, \ldots, \mathcal{U}_M$ with query sets $\mathcal{Q}_u$. A unit stores a candidate key block I_u; query i in unit $u(i)$ has causal limit f_i and active visible key set*

$$I_i^{\mathrm{vis}} = \{j \in I_{u(i)} : j \le f_i\}.$$

Let $\bar{W} = \frac{1}{N}\sum_{i=1}^{N} |I_i^{\mathrm{vis}}|$ denote the trajectory-average visible state length. Then, standard causal attention evaluates:

$$\mathcal{E}_{\mathrm{trace}} = \sum_{i=1}^{N} i = \frac{N(N+1)}{2} = \Theta(N^2) \tag{11}$$

pairwise query–key dot-products, whereas Replay Attention evaluates:

$$\mathcal{E}_{\mathrm{replay}} = \sum_{u=1}^{M} \sum_{i \in \mathcal{Q}_u} |I_i^{\mathrm{vis}}| = \sum_{i=1}^{N} |I_i^{\mathrm{vis}}| = N \cdot \bar{W} = \Theta(N \cdot \bar{W}). \tag{12}$$

Furthermore, if the visible route is capacity-bounded such that $|I_i^{\mathrm{vis}}| \le W_{\max}^{\mathrm{vis}} \le \infty$ for every query i (including fixed task context and any partially emitted update prefix), then $\bar{W} \le W_{\max}^{\mathrm{vis}} = \mathcal{O}(1)$ with respect to trajectory horizon T, and algorithmic attention complexity scales linearly with total trajectory length:

$$\mathcal{E}_{\mathrm{replay}} = \mathcal{O}(N) \quad \textit{versus} \quad \mathcal{E}_{\mathrm{trace}} = \Theta(N^2). \tag{13}$$

Proof. In standard causal self-attention over the flattened historical tape, query $i \in \{1, \ldots, N\}$ attends causally to all preceding keys $\{1, \ldots, i\}$, yielding $\sum_{i=1}^{N} i = N(N+1)/2$ non-zero attention matrix entries. Under Replay Attention, query event i evaluates attention over I_i^{vis}, the causal subset of its unit's candidate block, so the unit contributes $\sum_{i \in \mathcal{Q}_u} |I_i^{\mathrm{vis}}|$ active attention pairs. The unit-level rectangle $|\mathcal{Q}_u| \cdot |I_u|$ is an upper bound rather than an identity: it is tight only when every query observes the entire candidate block. For a growth prefix with an initial candidate context of size K_0, the newly emitted keys contribute $LK_0 + L(L+1)/2$ active pairs, while a rectangle would charge $L(K_0 + L)$. Summing over units yields $\mathcal{E}_{\mathrm{replay}} = \sum_{i=1}^{N} |I_i^{\mathrm{vis}}| = N \cdot \bar{W}$, and the visible-route bound gives $\bar{W} \le W_{\max}^{\mathrm{vis}}$ whenever the complete visible workspace envelope remains bounded, completing the proof. □

Measured at the matched validation frontiers, Replay Attention evaluates 56% (ListOps Reduction), 30% (DP-VE), and 13% (Typed λ-Calculus) of Transition Trace's active attention pairs under the no-refresh schedule, rising to 78%, 46%, and 14% at the default refresh horizon $\rho = 10$ (Table 1; operator execution latency is profiled separately in Appendix C).

B.4 Parallel Replay Execution Model and Replay Units

Replay Attention compiles the entire expert trajectory τ_T^* into an append-only physical tape $\mathcal{T}_{\mathrm{phys}}$ containing N unique token events. Each query $i \in \{1, \ldots, N\}$ is assigned an execution route $r_i = (I_{u(i)}, \boldsymbol{\ell}_{u(i)}, p_i, f_i)$, where:

1. $I_{u(i)} \subseteq \{1, \ldots, N\}$ is the candidate physical key block for the replay unit containing query i;
2. $\boldsymbol{\ell}_{u(i)} \in \mathbb{N}^{|I_{u(i)}|}$ specifies the unit-specific logical coordinates of the candidate keys;
3. $p_i \in \mathbb{N}$ denotes the logical coordinate of query i;
4. $f_i \in \{1, \ldots, N\}$ enforces the causal horizon, ensuring query i cannot attend to physical keys committed in future online steps ($j \le f_i$).

The query-visible set is $I_i^{\mathrm{vis}} = \{j \in I_{u(i)} : j \le f_i\}$.

Definition of Replay Unit. Naive implementation of query-dependent attention routes would degrade into irregular, token-by-token memory gathers. Replay Attention partitions the N query events into *replay units* so that shared candidate blocks and ragged causal schedules can be reused; the current kernel's measured latency is reported separately in Appendix C.3. A replay unit $u = (\mathcal{Q}_u, I_u, \boldsymbol{\ell}_u)$ is a maximal contiguous span of query events $\mathcal{Q}_u = \{i_{\text{start}}, \ldots, i_{\text{end}}\}$ such that:

1. All queries in $\mathcal{Q}_u$ reference the same ordered candidate key block $I_u \subseteq \{1, \ldots, N\}$;
2. The logical coordinate mapping $\boldsymbol{\ell}_u$ for that candidate block is constant;
3. Query logical positions advance sequentially: $p_{i+1} = p_i + 1$;
4. The unit's tile schedule and causal-limit representation remain valid for the query-specific limits f_i (fully future tiles may be omitted, while boundary tiles retain an elementwise causal mask).

Unit Boundary Triggers. A replay unit is terminated, and a new unit initiated, whenever any of the following events occur:

- **Active Span Mutation**: An edit transaction completes, excising target span O_t and publishing replacement V_t;
- **Coordinate Realignment**: Suffix keys undergo displacement $\Delta_t \neq 0$;
- **Boundary Refresh**: The post-commit refresh horizon initiates a localized re-encoding pass over Q_t^{ref}.

Within each replay unit, queries share the candidate key block I_u under query-specific causal limits f_i, skipping fully future tiles and applying an elementwise causal mask on boundary tiles (Figure 2).

B.5 Proposition 1: Algebraic Equivalence and Degeneration

Proposition 1 (Exact-Arithmetic Equivalence). *Suppose online execution and Replay Attention satisfy the following identification conditions for every event i in trajectory τ_T^*:*

1. ***Route Faithful Realization**: For query i corresponding to online step (t, k), I_i^{vis} indexes precisely the active physical key entries in $\mathcal{K}_t^* \cup \{u_{t,<k}^*\}$, the logical coordinates $\boldsymbol{\ell}_{u(i)}$ and p_i match the online coordinates $(\boldsymbol{\ell}_t^*, \ell_{t,k}^*)$, and f_i excludes future online events.*
2. ***Deterministic Transformer Block**: All projections, Q/K normalization, RoPE operators, GQA head sharing, feed-forward operators, residuals, and layer-normalization operators are identical and evaluated under exact arithmetic.*

Then for every layer $l \in \{1, \ldots, L\}$ and every event i, the intermediate activations and output logits coincide identically:

$$\mathbf{h}_i^{(l),\text{replay}} \equiv \mathbf{h}_{t,k}^{(l),\text{online}}, \qquad \hat{\mathbf{y}}_i^{\text{replay}} \equiv \hat{\mathbf{y}}_{t,k}^{\text{online}}. \tag{14}$$

Proof by Induction over Transformer Layers. We proceed by induction over Transformer layer depth $l \in \{0, 1, \ldots, L\}$.

Base Case ($l = 0$): At the input embedding layer, the representation is determined solely by the token identity and embedding tables:

$$\mathbf{h}_i^{(0),\text{replay}} = \mathbf{E}_{x_i} = \mathbf{E}_{x_{t,k}} = \mathbf{h}_{t,k}^{(0),\text{online}}. \tag{15}$$

Since token event i physically transcribes token $x_{t,k}$ by construction, the base case holds with exact identity.

Inductive Step ($l - 1 \to l$): Assume the inductive hypothesis holds at layer $l - 1$: for all physical events $j \in \{1, \ldots, N\}$ corresponding to online steps $(s(j), m(j))$,

$$\mathbf{h}_j^{(l-1),\text{replay}} = \mathbf{h}_{s(j),m(j)}^{(l-1),\text{online}}. \tag{16}$$

We now show that the output of layer l is identical for query event $i \sim (t, k)$.

1. **Pre-Attention Normalization and Projections**: Both implementations apply identical pre-attention RMSNorm operators to the incoming hidden state:

$$\bar{\mathbf{h}}_i^{(l-1),\text{replay}} = \text{RMSNorm}(\mathbf{h}_i^{(l-1),\text{replay}}) = \text{RMSNorm}(\mathbf{h}_{t,k}^{(l-1),\text{online}}) = \bar{\mathbf{h}}_{t,k}^{(l-1),\text{online}}. \tag{17}$$

Let $\mathcal{N}_q^{(l)}$ and $\mathcal{N}_k^{(l)}$ denote the model's query/key normalization operators (identity when a backbone does not expose them). The unrotated query and key vectors evaluate as:

$$\mathbf{q}_i = \mathcal{N}_q^{(l)}\Big(\mathbf{W}_Q^{(l)}\bar{\mathbf{h}}_i^{(l-1),\text{replay}}\Big) = \mathcal{N}_q^{(l)}\Big(\mathbf{W}_Q^{(l)}\bar{\mathbf{h}}_{t,k}^{(l-1),\text{online}}\Big) = \mathbf{q}_{t,k}. \tag{18}$$

Similarly, for every visible key $j \in I_i^{\text{vis}}$, the unrotated key and value vectors satisfy:

$$\mathbf{k}_j = \mathcal{N}_k^{(l)}\Big(\mathbf{W}_K^{(l)}\bar{\mathbf{h}}_j^{(l-1),\text{replay}}\Big) = \mathcal{N}_k^{(l)}\Big(\mathbf{W}_K^{(l)}\bar{\mathbf{h}}_{s(j),m(j)}^{(l-1),\text{online}}\Big) = \mathbf{k}_{s(j),m(j)}, \tag{19}$$

$$\mathbf{v}_j = \mathbf{W}_V^{(l)}\bar{\mathbf{h}}_j^{(l-1),\text{replay}} = \mathbf{W}_V^{(l)}\bar{\mathbf{h}}_{s(j),m(j)}^{(l-1),\text{online}} = \mathbf{v}_{s(j),m(j)}. \tag{20}$$

2. **Rotary Position Embeddings**: In online execution, query $\mathbf{q}_{t,k}$ is rotated by logical position $p_i = \ell_{t,k}^*$, while key $\mathbf{k}_{s(j),m(j)}$ in active context is rotated by its logical position $\ell_{u(i)}(j) = \ell_{t,j}^*$:

$$\tilde{\mathbf{q}}_{t,k} = \mathbf{R}_{p_i}\mathbf{q}_{t,k}, \qquad \tilde{\mathbf{k}}_{s(j),m(j)} = \mathbf{R}_{\ell_{u(i)}(j)}\mathbf{k}_{s(j),m(j)}. \tag{21}$$

In Replay Attention, the kernel synthesizes $\mathbf{R}_{p_i}$ and $\mathbf{R}_{\ell_{u(i)}(j)}$ on the fly within SRAM tiles:

$$\tilde{\mathbf{q}}_i = \mathbf{R}_{p_i}\mathbf{q}_i = \tilde{\mathbf{q}}_{t,k}, \qquad \tilde{\mathbf{k}}_{i,j} = \mathbf{R}_{\ell_{u(i)}(j)}\mathbf{k}_j = \tilde{\mathbf{k}}_{s(j),m(j)}. \tag{22}$$

3. **Attention Logits and Softmax Distribution**: For any visible key $j \in I_i^{\text{vis}}$, the attention score is:

$$A_{i,j} = \frac{\tilde{\mathbf{q}}_i^\top \tilde{\mathbf{k}}_{i,j}}{\sqrt{d_k}} = \frac{\tilde{\mathbf{q}}_{t,k}^\top \tilde{\mathbf{k}}_{s(j),m(j)}}{\sqrt{d_k}} = A_{t,k,j}^{\text{online}}. \tag{23}$$

For all inactive or future keys $j \notin I_{u(i)}$ or $j > f_i$, visibility masking sets the logit to $-\infty$. Because I_i^{vis} indexes precisely the active keys in $\mathcal{K}_t^* \cup \{u_{t,<k}^*\}$, the denominator of the softmax normalizer is algebraically identical:

$$\sum_{j \in I_i^{\text{vis}}} \exp(A_{i,j}) \equiv \sum_{j' \in \text{ActiveContext}_{t,k}} \exp(A_{t,k,j'}^{\text{online}}). \tag{24}$$

Consequently, the attention probability vectors $\boldsymbol{\alpha}_i^{\text{replay}}$ and $\boldsymbol{\alpha}_{t,k}^{\text{online}}$ match exactly.

4. **Value Aggregation, Residuals, and MLP Blocks**: The context aggregation vector evaluates to:

$$\mathbf{o}_i = \sum_{j \in I_i^{\text{vis}}} \alpha_{i,j}\mathbf{v}_j = \sum_{j' \in \text{ActiveContext}_{t,k}} \alpha_{t,k,j'}\mathbf{v}_{j'} = \mathbf{o}_{t,k}^{\text{online}}. \tag{25}$$

Passing through linear projection $\mathbf{W}_O$, residual connection, post-attention RMSNorm, and SwiGLU MLP:

$$\begin{aligned}\mathbf{h}_i^{(l),\text{replay}} &= \mathbf{h}_i^{(l-1),\text{replay}} + \mathbf{W}_O\mathbf{o}_i \\ &\quad + \text{MLP}\Big(\text{RMSNorm}\Big(\mathbf{h}_i^{(l-1),\text{replay}} + \mathbf{W}_O\mathbf{o}_i\Big)\Big) \equiv \mathbf{h}_{t,k}^{(l),\text{online}}.\end{aligned} \tag{26}$$

By induction, hidden representations match across all layers $l \in \{1, \ldots, L\}$. Applying the shared language modeling head $\mathbf{W}_{\text{LM}}$ establishes exact logit equivalence: $\hat{\mathbf{y}}_i^{\text{replay}} \equiv \hat{\mathbf{y}}_{t,k}^{\text{online}}$. □

Invariance to Online KV Slot Permutation and Periodic Compaction. During online inference, committing transition S_{t+1} does not require shifting retained suffix tensors in physical memory on every step. Under the two-tier cache lifecycle (Appendix F.4), an ordinary tail commit crops temporary control tokens from the transaction tail, leaves the newly materialized entries for V_t (and refreshed prefix Q_t^{ref}) appended at the physical tail, and masks superseded entries in O_t (and prior Q_t^{ref}) in place as tombstones ($V_{q,j} = 0$, i.e., additive logit mask $-\infty$). The resulting uncompacted physical cache layout is

$$\mathcal{C}_{t+1}^{\mathrm{raw}} = P_t \parallel O_t^{\dagger} \parallel (Q_t^{\mathrm{ref}})^{\dagger} \parallel Q_t^{\mathrm{keep}} \parallel V_t \parallel Q_t^{\mathrm{ref}}, \tag{27}$$

where $Q_t^{\mathrm{keep}} = Q_t[\min(|Q_t|, \rho) :]$ and $(\cdot)^{\dagger}$ denotes masked tombstone slots. When accumulated tombstones reduce active slot utilization $|\mathcal{K}_{t+1}|/|\mathcal{C}_{t+1}^{\mathrm{raw}}|$ below a threshold τ_{compact} (0.70 in our runtime), a compaction pass gathers active entries into contiguous logical order,

$$\mathcal{C}_{t+1}^{\mathrm{compact}} = P_t \parallel V_t \parallel Q_t^{\mathrm{ref}} \parallel Q_t^{\mathrm{keep}}, \tag{28}$$

reclaiming all tombstone slots without recomputing hidden states. Because each cache entry stores its unrotated vectors $(\mathbf{k}_j, \mathbf{v}_j)$ paired with its logical coordinate $\ell_{t+1}(j)$, rotary position injection $\tilde{\mathbf{k}}_j = \mathbf{R}_{\ell_{t+1}(j)}\mathbf{k}_j$ depends solely on $\ell_{t+1}(j)$ rather than the physical slot index. For any query q with active visible set $\mathcal{A}_q = \{j : V_{q,j} = 1\}$, tombstoned slots contribute $\exp(-\infty) = 0$ to the softmax normalizer, and for any permutation $\pi : \mathcal{A}_q \to \mathcal{A}_q$ of the physical storage slots, commutativity and associativity of summation yield

$$\mathbf{o}_q = \frac{\sum_{j\in\mathcal{A}_q} \exp\big(\tilde{\mathbf{q}}_q^{\top}\mathbf{R}_{\ell_{t+1}(\pi(j))}\mathbf{k}_{\pi(j)}/\sqrt{d_k}\big)\,\mathbf{v}_{\pi(j)}}{\sum_{j'\in\mathcal{A}_q} \exp\big(\tilde{\mathbf{q}}_q^{\top}\mathbf{R}_{\ell_{t+1}(\pi(j'))}\mathbf{k}_{\pi(j')}/\sqrt{d_k}\big)} \equiv \frac{\sum_{j\in\mathcal{A}_q} \exp\big(\tilde{\mathbf{q}}_q^{\top}\tilde{\mathbf{k}}_j/\sqrt{d_k}\big)\,\mathbf{v}_j}{\sum_{j'\in\mathcal{A}_q} \exp\big(\tilde{\mathbf{q}}_q^{\top}\tilde{\mathbf{k}}_{j'}/\sqrt{d_k}\big)}. \tag{29}$$

Consequently, KV rearrangement between $\mathcal{C}_{t+1}^{\mathrm{raw}}$ and $\mathcal{C}_{t+1}^{\mathrm{compact}}$ leaves attention outputs invariant under exact arithmetic (differing only in floating-point reduction order) and preserves Proposition 1.

Degeneration to Standard Autoregressive Attention. When the transition operator is restricted to monotonic prefix extension:

$$X_{t+1} = X_t \parallel V_t, \quad O_t = \emptyset, \quad \rho = 0, \tag{30}$$

no tokens are ever excised ($\mathbf{V}_{q,k} = 1$ for all preceding tokens) and suffix coordinates remain static ($\Delta_t = 0$). Under these conditions:

1. The query-visible key set I_i^{vis} simplifies to $\{1, \ldots, i\}$;
2. Logical coordinates coincide with physical tape indices: $\ell_{u(i)}(j) = j$ and $p_i = i$;
3. All queries merge into a single, monolithic replay unit spanning the entire sequence length N.

The Replay query route matrix degenerates strictly into the standard lower-triangular causal attention visibility mask $\mathbf{V} \in \{0,1\}^{N\times N}$ with $V_{i,j} = \mathbf{1}_{j\le i}$. Semantically, Replay Attention is then identical to standard causal self-attention; an implementation may still incur different scheduling overhead unless it dispatches the dense-prefix fast path.

C Systems Qualification and Execution-Cost Boundary

This appendix qualifies the current Replay Attention operator at the systems boundary. It answers four separate questions: what is measured, whether the operator computes the intended attention, what the current implementation costs, and which conclusions those measurements support. The routing and equivalence results in Appendix B establish the mechanism itself; here we characterize its behavior at the systems boundary.

Measurements use matched physical token lengths when comparing Replay with a dense reference; table captions give the device, software stack, precision, and aggregation rule.

Table 9: **Operator numerical qualification against an independent dense FP64 oracle.** Measured on an A800 (SM80) under the Qwen3-8B attention geometry (32 query heads, 8 KV heads, $d_k = 128$). Maxima over two mechanism cases (ragged physical causality, logical-state repositioning) and three seeds each; candidate and oracle receive identical inputs, parameters, logical positions and upstream gradients. The two panels report relative L_2 error and maximum absolute error, respectively. Relative error is $\|x_{\text{cand}} - x_{\text{ref}}\|_2 / \|x_{\text{ref}}\|_2$. The strict FP32 IEEE path stays at 10^{-7}; the BF16 row is the production rounding envelope, a few tenths of a percent.

Relative L_2 error

Path	Dtype / device	O	dQ	dK	dV
Triton (IEEE)	FP32 / SM80	9.27×10^{-8}	3.02×10^{-7}	3.07×10^{-7}	1.25×10^{-7}
Triton	BF16 / SM80	2.10×10^{-3}	3.45×10^{-3}	3.77×10^{-3}	2.85×10^{-3}

Maximum absolute error

Path	Dtype / device	O	dQ	dK	dV
Triton (IEEE)	FP32 / SM80	3.33×10^{-7}	6.22×10^{-7}	8.20×10^{-7}	7.30×10^{-7}
Triton	BF16 / SM80	9.48×10^{-3}	9.98×10^{-3}	1.08×10^{-2}	2.18×10^{-2}

C.1 Scope and Measurement Contract

The paper uses three related but non-interchangeable quantities:

1. **Processed input tokens**: the tokens actually passed through the model across a training run. This is the cost axis used for the learning-curve controls in Appendix A.4; it is distinct from the number of supervised target tokens.
2. **Active attention pairs**: the route-visible query–key pairs admitted by a compiled Replay route, $\mathcal{E}_{\text{replay}} = \sum_i |I_i^{\text{vis}}|$ (Proposition B.3). This measures algorithmic attention complexity, distinct from hardware instruction count or wall-clock latency.
3. **Operator latency**: the measured forward, backward, or combined time of the attention wrapper, including its applicable dispatch and autograd work but excluding model projections, MLP layers, the optimizer, and unrelated training overhead.

The current prototype does **not yet yield net wall-clock speedup** over mature fused dense causal kernels at matched physical lengths. The systems measurements therefore separate the algorithmic reduction in active attention pairs from the execution cost of the present Triton implementation. They should not be read as an end-to-end throughput result or as a replacement for the statistical learning comparisons in Section 4.

C.2 Numerical and Route Qualification

The operator stores physical keys without pre-applied logical RoPE, synthesizes the required logical coordinates inside the tiled attention loop, and accumulates gradients for physical keys that are referenced by multiple routes through atomic or segmented reductions. These mechanisms are the implementation counterpart of the routing model formalized in Appendix B.

A numerical qualification compares identical inputs, parameters, logical positions, and upstream gradients against an independent dense FP64 oracle. We report relative L_2 error, maximum absolute error, and finite-value checks. Table 9 covers the Qwen3-8B attention geometry, two mechanism cases, and three seeds per case. The strict FP32 IEEE path remains at roughly 10^{-7} relative error, while the production BF16 path remains in the few-tenths-of-a-percent range for the output and all three gradients.

Route identification is qualified separately from floating-point error on frozen runtime traces covering horizons 1, 4, 16, and 25. Across $4{,}322$ queries and 46 commit events, the replay routes matched the online routes in query identity, visible-key set, logical coordinates, lifecycle, and causal limits, in agreement with the equivalence result of Appendix B.

Table 10: **Workload accounting of the ListOps Reduction validation episodes.** Computed on CPU from the compiled execution plans of every validation episode in each release (48 episodes: four shape strata of 12), with no GPU timing. "Tape tokens" is the shared physical length of the Theseus-AR and Trace realizations of the same episode; "units" counts the replay units of the Theseus-AR tape, and "edges" counts the visible query–key pairs each realization evaluates. The reduction column is the share of dense causal edges that Replay Attention never evaluates.

Budget n	Episodes	Tape tokens (median [min–max])	Units	Replay edges	Dense edges	Reduction
29	48	917 [830–995]	70	282,520	420,903	33%
58	48	1,668 [1,496–1,813]	137	831,020	1,391,946	40%
84	48	2,344 [2,091–2,578]	197	1,544,623	2,749,513	44%
112	48	3,068 [2,731–3,308]	261	2,583,788	4,709,383	45%

Table 11: **Replay Attention operator cost on the ListOps Reduction validation episodes.** Operator-level wrapper latency (autograd included) at each episode's own physical length, measured on an A800 (SM80) with PyTorch 2.12.0+cu130, CUDA 13.0 and Triton 3.7.0, under the attention geometry shared by both backbones (32 query heads, 8 KV heads, $d_k = 128$), BF16 post-RoPE inputs and the scanned causal schedule. Five repeats of 50 timed iterations after 20 warmup calls; every row is the median over the four shape strata of the release, one validation episode each. The dense reference is the fused SDPA kernel at the same physical token count, so token length is matched while the visible-edge counts differ by construction (Table 10). The locally adapted Triton FlashAttention-2 rows cover only the budgets inside its 2,048-token causal bound; its ratio is taken against the Replay reading measured in the same paired round.

Budget n	Implementation	Forward (ms)	Backward (ms)	Combined (ms)	Replay ÷ reference
29	Dense SDPA (fused)	0.239	0.453	0.753	5.99×
	Replay Attention	0.352	4.159	4.512	—
	Triton FlashAttention-2 (local)	0.336	0.691	1.015	4.44×
58	Dense SDPA (fused)	0.459	0.815	1.274	8.11×
	Replay Attention	1.006	9.334	10.334	—
	Triton FlashAttention-2 (local)	0.648	1.269	1.915	5.41×
84	Dense SDPA (fused)	0.683	1.297	1.991	9.00×
	Replay Attention	1.799	16.126	17.918	—
112	Dense SDPA (fused)	0.948	1.923	2.893	10.03×
	Replay Attention	2.961	26.049	29.008	—

C.3 Workload and Measured Execution Cost

The primary cost measurements use the ListOps Reduction validation episodes. For each operator-node budget $n \in \{29, 58, 84, 112\}$, Replay and Trace materialize the same episode and therefore carry the same physical token count. Replay evaluates its compiled route while Trace uses the dense causal path, so the comparison separates active-pair reduction from the cost of the operator implementation.

Table 10 aggregates all 48 validation episodes per release. The number of replay units grows with the semantic trajectory: from 70 units and 917 physical tokens at $n = 29$ to 261 units and 3,068 tokens at $n = 112$. The compiled route skips 33%–45% of dense causal edges across these releases.

Table 11 reports wrapper latency for one median-length episode per shape stratum. Replay is slower than fused dense attention at every budget: 6.0×, 8.1×, 9.0×, and 10.0× at $n = 29, 58, 84, 112$, respectively. The forward ratio ranges from 1.5× to 3.1×, whereas the backward ratio ranges from 9.2× to 13.5×; the measurements therefore locate the current gap primarily at the phase level, not at a single kernel component. The local Triton FlashAttention-2 rows provide an additional implementation reference only within its 2,048-token causal bound.

C.3.1 Dense-Prefix Boundary Case

As a boundary case, we also measure a fixed-length synthetic tape whose leading replay unit carries most of the visible references and whose state does not contract. Replay is then only 2.6× to 3.9×

Table 12: **Dense-prefix limit: Replay Attention on a fixed-length synthetic construction tape.** The tape is one action of a synthetic eight-node construction generator padded with unsupervised filler tokens to the requested physical length, so its replay units never contract: the leading unit alone carries the fraction of visible edge references listed below, which makes this the regime in which Replay Attention has least to save. Measured on an A800 (SM80), PyTorch 2.12.0+cu130, CUDA 13.0, Triton 3.7.0, BF16 post-RoPE inputs and the scanned causal schedule; median forward+backward wrapper latency over five repeats of 50 timed iterations after 20 warmup calls, dense and Replay alternating within one round. The dense reference is the native fused SDPA kernel at the same physical token count.

Physical tokens	Filler units	Leading-unit edge share	Replay edges	Dense edges	Dense (ms)	Replay (ms)	Replay ÷ dense
$N = 1{,}024$	363	79.2%	524,787	524,800	0.775	2.043	2.64×
$N = 2{,}048$	1,387	89.3%	2,098,163	2,098,176	1.614	5.354	3.32×
$N = 4{,}096$	3,435	94.6%	8,390,643	8,390,656	4.456	16.259	3.65×
$N = 8{,}192$	7,531	—	33,558,515	33,558,528	14.192	54.871	3.87×

slower than dense attention from 1K to 8K tokens, with an edge count within 0.01% of dense causal attention. The tape fills a length budget rather than representing a solved task trajectory, so it is reported only as a dense-prefix limit.

C.4 Interpretation and Limitations

The evidence supports three bounded conclusions. First, Replay Attention removes a substantial fraction of active attention pairs on multi-step task trajectories. Second, the current Triton prototype does not convert that reduction into wall-clock speedup against fused dense attention under the reported conditions. Third, the excess is concentrated in backward execution, but total latency alone does not establish a component-level attribution. Scheduling overhead, irregular gathers, dynamic RoPE arithmetic, and shared-key gradient reduction are plausible contributors, not separately measured causes.

Accordingly, this appendix establishes an operator-level numerical and cost boundary for the current implementation. Kernel fusion, block-sparse staging, coordinate pre-baking, and other optimization directions remain engineering work rather than results claimed here.

D Comprehensive Stepwise Transition Evaluation Records

This appendix provides the complete tabular records and metric contracts for the Stepwise Transition Evaluation protocol along Oracle trajectories summarized in Section 4.3.

All three task tables use the same column contract: *Condition*, *Bucket/Scale*, *Episodes*, *Steps*, strict error, legal error, failure-free horizon (FFH), and truncation count. A dash denotes a quantity that was not collected for that task; it is not a zero. Lambda-specific strict/legal zero-error counts are omitted here so that the three task tables have identical columns.

Each task is also reported as a complete curve set: the full ListOps Reduction scale sweep, the DP-VE 4k frontier, and the Typed λ-Calculus fixed4 buckets. The corresponding two-scale and two-bucket excerpt used by the main text is a separate figure drawn from the same runs.

D.1 Evaluation Metric Contracts

All Oracle-conditioned rollouts evaluate candidate transactions along ground-truth trajectory states S_t^* without compounding syntax errors:

- **Strict Action Error (%)**: Fraction of transitions where the generated edit transaction does not identically match the canonical teacher token sequence ($U_t \neq U_t^*$).
- **Legal Action Error (%)**: Fraction of transitions where the candidate transaction fails the deterministic domain verifier: $\mathrm{Valid}(S_t^*, \hat{U}_t) = 0$. Valid alternative execution paths (such as commuting subtree reductions or alternative variable selections) are counted as legal successes.

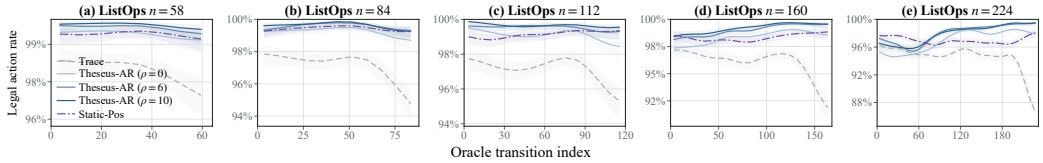


Figure 12: **Complete ListOps Reduction stepwise action validity across the operator-node scales.** Legal action rate against Oracle transition index for all five operator budgets ($n \in \{58, 84, 112, 160, 224\}$), comparing Transition Trace, Theseus-AR ($\rho = 0, 6, 10$), and the Static-Pos control. Shaded bands show descriptive pointwise Gaussian-weighted local Wilson intervals over evaluation steps (paired episode-cluster bootstrap CIs accounting for within-episode correlation are reported in Table 2); Section 4.3 reports the $n = 58$ and $n = 224$ excerpt only.

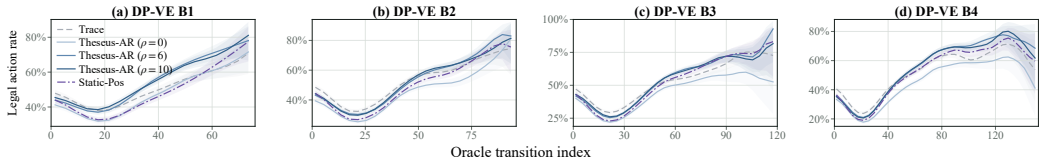


Figure 13: **Complete DP-VE stepwise action validity on the shared 4k frontier.** Legal action rate against Oracle transition index for the four difficulty buckets ($B1$–$B4$), comparing Transition Trace, Theseus-AR ($\rho = 0, 6, 10$), and the Static-Pos control on the same sample and protocol as Table 15. Shaded bands show descriptive pointwise local Wilson intervals over evaluation steps (paired episode-cluster bootstrap CIs are in Table 2); Section 4.3 reports the $B1$ and $B4$ excerpt only, and the 8k continuation is reported as a table without a matched curve set.

- **Failure-Free Horizon (FFH)**: The mean continuous segment length before the first illegal transition is generated. Truncated episodes are accounted for without inflating the denominator.
- **Zero-Error Oracle Episodes**: The percentage of evaluation episodes in which every Oracle-anchored step across the trajectory incurs zero strict or legal error (a step-validity aggregate rather than a free-running rollout completion rate; Appendix E).

D.2 ListOps Reduction Oracle-Scan Ladder

Table 13 provides the complete evaluation ladder across operator-node budgets $n \in \{58, 84, 112, 160, 224\}$. While Transition Trace exhibits progressive error accumulation as prefix history expands (legal error reaching 5.37% at $n = 224$), Theseus-AR maintains legal error between 0.22% and 2.11%.

Table 14 records the follow-up evaluation of the State-Recompute materializer. Because this pass evaluates oracle-exact actions without the offline legal-alternative verdict stage, its strict error is reported separately and is not directly comparable to legal error estimates.

D.3 DP-VE Oracle-Scan Ladders

Table 15 reports the complete four-bucket matrix on the 4k frontier that the main text and the Static-Pos control share (800 episodes); all five conditions are evaluated on the same sample under the same protocol. Static-Pos is the worst condition overall (59.97% pooled legal error), above both Trace (56.52%) and Theseus-AR ($\rho = 10$, 57.10%), matching its training-side ordering.

Table 16 reports the complete four-bucket evaluation matrix (800 episodes) for DP-VE under the 8k continuation. In deep buckets (B3, B4), Theseus-AR ($\rho \in \{6, 10\}$) demonstrates superior failure-free horizons and lower strict/legal error.

Table 13: **Complete ListOps Reduction Stepwise Transition Evaluation records.** Strict and legal action errors are percentages; legal alternatives are counted only in the legal column. FFH and truncation were not recorded for ListOps Reduction and are shown as –.

Condition	Bucket/Scale	Episodes	Steps	Strict Err. ↓	Legal Err. ↓	FFH ↑	Trunc.
Transition Trace	$n = 58$	100	5,800	2.017	2.017	–	–
Transition Trace	$n = 84$	100	8,400	2.905	2.905	–	–
Transition Trace	$n = 112$	100	11,200	2.821	2.804	–	–
Transition Trace	$n = 160$	100	16,000	3.887	3.863	–	–
Transition Trace	$n = 224$	100	22,400	5.397	5.375	–	–
Transition Trace	All	500	63,800	3.931	3.914	–	–
Static-Pos ($\rho = 10$)	$n = 58$	100	5,800	0.586	0.586	–	–
Static-Pos ($\rho = 10$)	$n = 84$	100	8,400	0.571	0.571	–	–
Static-Pos ($\rho = 10$)	$n = 112$	100	11,200	0.857	0.857	–	–
Static-Pos ($\rho = 10$)	$n = 160$	100	16,000	1.694	1.687	–	–
Static-Pos ($\rho = 10$)	$n = 224$	100	22,400	3.067	3.054	–	–
Static-Pos ($\rho = 10$)	All	500	63,800	1.781	1.774	–	–
Theseus-AR ($\rho = 0$)	$n = 58$	100	5,800	0.603	0.586	–	–
Theseus-AR ($\rho = 0$)	$n = 84$	100	8,400	0.750	0.750	–	–
Theseus-AR ($\rho = 0$)	$n = 112$	100	11,200	0.884	0.875	–	–
Theseus-AR ($\rho = 0$)	$n = 160$	100	16,000	1.687	1.675	–	–
Theseus-AR ($\rho = 0$)	$n = 224$	100	22,400	3.071	3.058	–	–
Theseus-AR ($\rho = 0$)	All	500	63,800	1.810	1.799	–	–
Theseus-AR ($\rho = 6$)	$n = 58$	100	5,800	0.345	0.328	–	–
Theseus-AR ($\rho = 6$)	$n = 84$	100	8,400	0.429	0.429	–	–
Theseus-AR ($\rho = 6$)	$n = 112$	100	11,200	0.455	0.455	–	–
Theseus-AR ($\rho = 6$)	$n = 160$	100	16,000	1.019	1.013	–	–
Theseus-AR ($\rho = 6$)	$n = 224$	100	22,400	**2.031**	**2.022**	–	–
Theseus-AR ($\rho = 6$)	All	500	63,800	1.136	1.130	–	–
Theseus-AR ($\rho = 10$)	$n = 58$	100	5,800	**0.259**	**0.224**	–	–
Theseus-AR ($\rho = 10$)	$n = 84$	100	8,400	**0.369**	**0.369**	–	–
Theseus-AR ($\rho = 10$)	$n = 112$	100	11,200	**0.339**	**0.339**	–	–
Theseus-AR ($\rho = 10$)	$n = 160$	100	16,000	**0.825**	**0.819**	–	–
Theseus-AR ($\rho = 10$)	$n = 224$	100	22,400	2.125	2.112	–	–
Theseus-AR ($\rho = 10$)	All	500	63,800	**1.085**	**1.075**	–	–

Table 14: **State-Recompute Stepwise Transition Evaluation on ListOps Reduction.** The oracle action error is computed from the State-Recompute evaluation; complete-trajectory strict success is reported separately. This pass did not produce an offline legal-alternative verdict, so no legal-error estimate is inferred.

Scale	Episodes	Actions	Action Err. ↓	Strict Zero	Rollout Actions	Peak GB
$n = 29$	100	2,900	34.448	0	999	8.40
$n = 58$	100	5,800	36.172	0	2,098	9.80
$n = 84$	100	8,400	38.310	0	3,218	11.49
$n = 112$	100	11,200	40.125	0	4,494	14.39

D.4 Typed λ-Calculus fixed4 Oracle-Scan Ladder

Table 17 summarizes the complete four-bucket evaluation (400 episodes per condition) for Typed λ-Calculus on the Qwen3-8B backbone. On the deep buckets (B3, B4), Theseus-AR ($\rho = 10$) achieves the lowest legal error (1.905% and 1.196%) with only a single truncated episode across all 400 rollouts, lowering pooled legal error from 1.637% (Transition Trace) to 1.463%.

Table 15: **Complete DP-VE (4k frontier) Stepwise Transition Evaluation records.** Records are the 4k frontier; the 8k continuation is Table 16. Strict and legal errors are semantic-step action error rates (%), with lower values better. FFH is the mean failure-free horizon; truncations remain in the action denominator.

Condition	Bucket/Scale	Episodes	Steps	Strict Err. ↓	Legal Err. ↓	FFH ↑	Trunc.
Transition Trace	B1	200	7,642	55.195	55.195	7.08	7
Transition Trace	B2	200	10,812	**55.281**	**55.281**	8.99	2
Transition Trace	B3	200	12,849	**57.063**	**57.063**	8.61	3
Transition Trace	B4	200	15,373	**57.607**	**57.607**	7.39	4
Transition Trace	All	800	46,676	**56.524**	**56.524**	8.02	16
Static-Pos ($\rho = 10$)	B1	200	7,642	59.474	59.474	7.33	7
Static-Pos ($\rho = 10$)	B2	200	10,812	60.063	60.063	7.77	7
Static-Pos ($\rho = 10$)	B3	200	12,849	60.386	60.386	7.82	14
Static-Pos ($\rho = 10$)	B4	200	15,373	59.793	59.793	6.33	20
Static-Pos ($\rho = 10$)	All	800	46,676	59.967	59.967	7.32	48
Theseus-AR ($\rho = 0$)	B1	200	7,642	60.102	60.102	5.79	21
Theseus-AR ($\rho = 0$)	B2	200	10,812	62.902	62.902	5.53	21
Theseus-AR ($\rho = 0$)	B3	200	12,849	63.608	63.608	5.74	20
Theseus-AR ($\rho = 0$)	B4	200	15,373	64.574	64.574	5.48	34
Theseus-AR ($\rho = 0$)	All	800	46,676	63.189	63.189	5.63	96
Theseus-AR ($\rho = 6$)	B1	200	7,642	54.423	54.423	6.34	1
Theseus-AR ($\rho = 6$)	B2	200	10,812	57.556	57.556	7.34	1
Theseus-AR ($\rho = 6$)	B3	200	12,849	58.386	58.386	6.54	3
Theseus-AR ($\rho = 6$)	B4	200	15,373	57.848	57.848	5.80	8
Theseus-AR ($\rho = 6$)	All	800	46,676	57.368	57.368	6.51	13
Theseus-AR ($\rho = 10$)	B1	200	7,642	**53.533**	**53.533**	6.32	2
Theseus-AR ($\rho = 10$)	B2	200	10,812	57.205	57.205	7.34	4
Theseus-AR ($\rho = 10$)	B3	200	12,849	57.989	57.989	6.38	6
Theseus-AR ($\rho = 10$)	B4	200	15,373	58.050	58.050	5.75	9
Theseus-AR ($\rho = 10$)	All	800	46,676	57.098	57.098	6.45	21

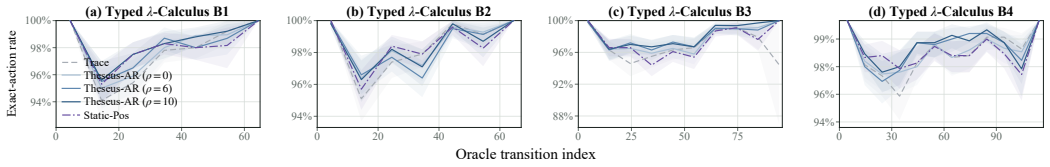


Figure 14: **Complete Typed λ-Calculus stepwise exact-action validity.** Teacher-forced exact-action rate against Oracle transition index for the four difficulty buckets ($B1$–$B4$), comparing Transition Trace, Theseus-AR ($\rho = 0, 6, 10$), and the Static-Pos control. The plotted quantity is the per-action strict exact-match rate, i.e. the strict column of Table 17: verdicts collected after a first mismatch are not comparable across evaluation modes, so the legal column is reported in the table only. Shaded bands are 95% episode-cluster bootstrap intervals.

E Segment-Level State-Anchored Evaluation Protocol

This appendix formalizes the *Stepwise Transition Evaluation* protocol and its segment-level teacher-forced rollout (TFR) implementation introduced in Section 4. By generating each candidate action from a canonical Oracle state S_t^* and verifying it with an exact domain checker before anchoring the next step at S_{t+1}^*, this protocol isolates stepwise transition competence across deep horizons without compounding syntax errors or false penalties from branching symmetries.

Table 16: **Complete DP-VE Stepwise Transition Evaluation records.** Strict and legal errors are semantic-step action error rates (%), with lower values better. FFH is the mean failure-free horizon; truncations remain in the action denominator.

Condition	Bucket/Scale	Episodes	Steps	Strict Err. ↓	Legal Err. ↓	FFH ↑	Trunc.
Transition Trace	B1	200	7,642	51.596	51.596	8.27	2
Transition Trace	B2	200	10,812	**52.321**	**52.321**	9.46	3
Transition Trace	B3	200	12,849	**53.654**	**53.654**	9.81	3
Transition Trace	B4	200	15,373	**54.791**	**54.791**	9.23	3
Transition Trace	All	800	46,676	**53.383**	**53.383**	9.19	11
Theseus-AR ($\rho = 0$)	B1	200	7,642	57.236	57.236	6.47	14
Theseus-AR ($\rho = 0$)	B2	200	10,812	59.036	59.036	7.55	12
Theseus-AR ($\rho = 0$)	B3	200	12,849	59.530	59.530	7.48	16
Theseus-AR ($\rho = 0$)	B4	200	15,373	60.476	60.476	6.42	17
Theseus-AR ($\rho = 0$)	All	800	46,676	59.352	59.352	6.98	59
Theseus-AR ($\rho = 6$)	B1	200	7,642	51.662	51.662	7.13	1
Theseus-AR ($\rho = 6$)	B2	200	10,812	55.253	55.253	8.85	1
Theseus-AR ($\rho = 6$)	B3	200	12,849	56.557	56.557	7.45	6
Theseus-AR ($\rho = 6$)	B4	200	15,373	56.489	56.489	6.88	9
Theseus-AR ($\rho = 6$)	All	800	46,676	55.431	55.431	7.58	17
Theseus-AR ($\rho = 10$)	B1	200	7,642	**50.196**	**50.196**	8.71	1
Theseus-AR ($\rho = 10$)	B2	200	10,812	54.624	54.624	9.47	3
Theseus-AR ($\rho = 10$)	B3	200	12,849	55.802	55.802	7.71	2
Theseus-AR ($\rho = 10$)	B4	200	15,373	55.578	55.578	8.02	7
Theseus-AR ($\rho = 10$)	All	800	46,676	54.538	54.538	8.47	13

E.1 Evaluation Methodology and Metric Contracts

Our empirical evaluation isolates architectural and representational mechanics from confounding data noise. All tasks satisfy four foundational properties:

1. **Deterministic Oracles**: Every transition step and terminal normal form is computed by an exact algorithmic solver, eliminating subjective prompting, human annotation noise, or judge LLM variance.
2. **Oracle-Conditioned State Anchoring**: At each Oracle transition state S_t^*, the model predicts a localized edit transaction and the deterministic verifier checks the resulting successor. The next evaluated action is reset to the next canonical state S_{t+1}^*, so a failed candidate does not alter later evaluation inputs, cleanly decoupling conditional transition competence from compounding exposure bias.
3. **Hardware-Agnostic Workload Accounting**: Compute is measured in cumulative input tokens (the sum of tokens processed across all forward passes). This eliminates cluster scheduling variance, thermal throttling, and memory-bandwidth jitter, directly reflecting the attention computational complexity ($\mathcal{O}(L^2)$ for monotonically expanding histories vs. $\mathcal{O}(L \cdot S)$ for in-place discrete state recurrence, where $S \ll L$ is active state size).
4. **Stepwise Metric Evaluation**: We report per-transition strict and legal validity. Strict validity requires the canonical action, whereas legal validity accepts any action certified by the deterministic domain verifier.

E.2 The Equivalence Dilemma: Unique vs. Non-Unique Transitions

Evaluating multi-step discrete recurrence under verifiable ground truth requires distinguishing between token-level sequence agreement and semantic action validity.

Lemma 1 (First-Divergence Equivalence Under Unique Transitions). *Let* $\tau^* = (S_0^*, U_0^*, S_1^*, \ldots, U_{T-1}^*, S_T^*)$ *be a canonical ground-truth transition trajectory. Suppose that*

Table 17: **Complete Typed λ-Calculus Stepwise Transition Evaluation records.** Strict and legal errors are action-level percentages; legal alternatives are accepted only in the legal column. FFH is the mean failure-free horizon; truncations remain in the action denominator. The Static-Pos control is rolled out on the same four buckets under the same protocol.

Condition	Bucket/Scale	Episodes	Steps	Strict Err. ↓	Legal Err. ↓	FFH ↑	Trunc.
Transition Trace	B1	100	5,385	2.136	2.136	33.55	1
Transition Trace	B2	100	5,777	0.848	0.848	45.25	0
Transition Trace	B3	100	8,136	2.077	2.077	47.25	2
Transition Trace	B4	100	10,705	1.476	1.476	56.31	0
Transition Trace	All	400	30,003	1.637	1.637	45.59	3
Static-Pos ($\rho = 10$)	B1	100	5,385	**1.616**	**1.616**	38.56	2
Static-Pos ($\rho = 10$)	B2	100	5,777	**0.519**	**0.519**	47.09	2
Static-Pos ($\rho = 10$)	B3	100	8,136	2.212	2.212	45.72	8
Static-Pos ($\rho = 10$)	B4	100	10,705	1.252	1.252	62.34	4
Static-Pos ($\rho = 10$)	All	400	30,003	**1.437**	**1.437**	48.43	16
Theseus-AR ($\rho = 0$)	B1	100	5,385	2.154	2.154	34.53	0
Theseus-AR ($\rho = 0$)	B2	100	5,777	1.021	0.987	45.29	0
Theseus-AR ($\rho = 0$)	B3	100	8,136	2.458	2.446	41.65	2
Theseus-AR ($\rho = 0$)	B4	100	10,705	1.616	1.616	57.87	0
Theseus-AR ($\rho = 0$)	All	400	30,003	1.826	1.816	44.84	2
Theseus-AR ($\rho = 6$)	B1	100	5,385	2.061	2.043	35.22	0
Theseus-AR ($\rho = 6$)	B2	100	5,777	1.073	1.073	43.45	3
Theseus-AR ($\rho = 6$)	B3	100	8,136	1.967	1.967	46.52	2
Theseus-AR ($\rho = 6$)	B4	100	10,705	1.401	1.383	57.72	0
Theseus-AR ($\rho = 6$)	All	400	30,003	1.610	1.600	45.73	5
Theseus-AR ($\rho = 10$)	B1	100	5,385	1.746	1.708	37.25	0
Theseus-AR ($\rho = 10$)	B2	100	5,777	1.125	1.108	44.25	0
Theseus-AR ($\rho = 10$)	B3	100	8,136	**1.917**	**1.905**	46.30	0
Theseus-AR ($\rho = 10$)	B4	100	10,705	**1.233**	**1.196**	62.01	1
Theseus-AR ($\rho = 10$)	All	400	30,003	1.490	1.463	47.45	1

at every intermediate state S_t^*, the set of semantically valid successor actions $\mathcal{U}(S_t^*)$ is a singleton ($|\mathcal{U}(S_t^*)| = 1$), uniquely materialized as token sequence $U_t^* = (u_{t,1}^*, \ldots, u_{t,m_t}^*)$. Then evaluating the model under teacher-forced canonical prefixes to identify the first divergent token is algebraically identical to free token-by-token closed-loop rollout failure:

$$\mathbb{P}_{\textit{closed-loop}}\big(\textit{Success up to step } t\big) \equiv \mathbb{P}_{\textit{teacher-forced}}\big(\forall t' \leq t,\ U_{t'} = U_{t'}^*\big). \tag{31}$$

Proof. In a deterministic environment with unique transitions, a generated trajectory remains on the valid manifold if and only if every emitted token matches the unique valid action string. If the model emits $u_{t,k} \neq u_{t,k}^*$, the unrolled sequence state immediately diverges from the valid state manifold. Because the successor is corrupted, all subsequent steps diverge. Thus, the index of the first mismatched token under teacher-forced canonical inputs coincides exactly with the point of failure in an unconstrained rollout. ∎

Failure of Token-Level Matching Under Branching Symmetries. In contrast to arithmetic or synthetic string copying, our structural tasks (ListOps Reduction, DP-VE, and Typed λ-Calculus) exhibit rich *concurrency and branching symmetries*:

- In **ListOps Reduction**, an expression tree may contain multiple disjoint ready subtrees $\omega_1(v_1, \ldots), \ldots, \omega_k(v_k, \ldots)$. Reducing ω_1 before ω_2 versus ω_2 before ω_1 produces identical normal forms and represents mathematically equivalent reduction strategies.
- In **DP-VE**, multiple independent clauses may satisfy the preconditions for unit propagation or variable resolution concurrently.

When $|\mathcal{U}(S_t^*)| > 1$, standard token-level teacher forcing enforces an arbitrary total ordering over mutually commute-able operations. If the model emits a valid transition $U_t \in \mathcal{U}(S_t^*)$ that differs from the oracle's arbitrary tie-breaking choice U_t^*, token-level matching falsely penalizes the model. Evaluating solely via token matching therefore introduces catastrophic false-negative biases.

Intra-Action Permutation Symmetries and Phase Boundaries. Beyond inter-action branching symmetries, structured sequence updates exhibit *intra-action permutation symmetries and internal attention phase transitions*. In DP-VE, for instance, an append transaction may introduce a set of independent resolvent clauses whose line-by-line serialization order is mathematically arbitrary. Forcing teacher evaluation against one arbitrary canonical clause order penalizes valid commutative permutations and triggers rollouts from ill-posed intermediate line slices. Furthermore, discrete recurrence updates undergo internal topological phase transitions: query tokens before <|with|> operate under the target-locator context, whereas queries after <|with|> operate under the replacement context with distinct key visibility and dynamic coordinate realignments. Faithful evaluation therefore requires decomposing updates into structured **segments**.

E.3 Hierarchical Formulation of Segment-Level State-Anchored Evaluation

To measure true policy capacity across deep horizons without token-matching artifacts or intra-action permutation penalties, we define the **Segment-Level Teacher-Forced Rollout (TFR)** protocol paired with hierarchical semantic verification.

Segment Decomposition and Delimiter Contracts. At decision boundary t, the candidate transition update is decomposed into an ordered sequence of execution segments $\boldsymbol{g}_t = (g_{t,1}, \ldots, g_{t,K_t})$. Each segment $g_{t,j}$ represents an unbroken autoregressive phase with distinct syntactic and topological responsibilities:

- **Atomic Transactions**: Standard in-place replacement or deletion operations are mapped to a single segment terminating at delimiter set $\{\texttt{<|commit|>}\}$.
- **Compound Append Transactions**: Multi-clause appends decompose into a target locator segment (terminating at $\{\texttt{<|with|>}\}$), followed by individual clause line segments (terminating at $\{\texttt{\textbackslash n}, \texttt{<|commit|>}\}$), and a terminal line segment (terminating at $\{\texttt{<|commit|>}\}$).

Each segment $g_{t,j}$ is governed by its own terminator delimiter set $\mathcal{T}_j$ and an admissible local content set $\mathcal{L}(S_t^*, j)$ (such as the multiset of remaining valid resolvent clauses), enabling order-invariant verification.

Segment-Level Visibility and Shadow Slot Isolation. Let canonical trajectory τ^* be compiled into an immutable physical tape under Replay Attention. For segment $g_{t,j}$, the active visibility set $\mathcal{V}(S_t^*, j)$ is instantiated via the query route at its canonical divergence anchor:

1. **Active Key Selection ($I_{t,j}$)**: Exposes precisely the physical entries corresponding to active tokens in state $S_t^* = P_t^* \parallel Q_t^*$ at that anchor. Excised prior spans $O_{<t}^*$ remain masked ($\mathbf{V}_{q,k} = 0$), where $\mathbf{V} \in \{0,1\}$ is the binary visibility mask.
2. **Dynamic Coordinate Realignment ($\boldsymbol{\ell}_{t,j}$)**: Suffix keys are rotated using their accumulated translation offsets at the anchor.
3. **Phase Invariance Across <|with|>**: When a segment transitions from target localization to replacement content, the visibility plan shifts the query anchor and active key selection to the replacement branch, preserving representational continuity without rematerializing preceding canonical prefixes.
4. **Causal Horizon Isolation ($f_{t,j}$)**: The causal frontier $f_{t,j}$ strictly forbids attending to any future canonical events or tokens ($t' > t$).
5. **Autoregressive Generation Masking**: During generation of the candidate segment string $g_{t,j} = (v_{j,1}, \ldots, v_{j,m_j})$, token $v_{j,k}$ attends causally to $\mathcal{V}(S_t^*, j) \cup \{v_{j,1}, \ldots, v_{j,k-1}\}$.
6. **Frozen Prefix and Shadow Slot Isolation**: Canonical prefix representations are strictly read-only throughout an episode. Generated tokens occupy dedicated shadow slots that

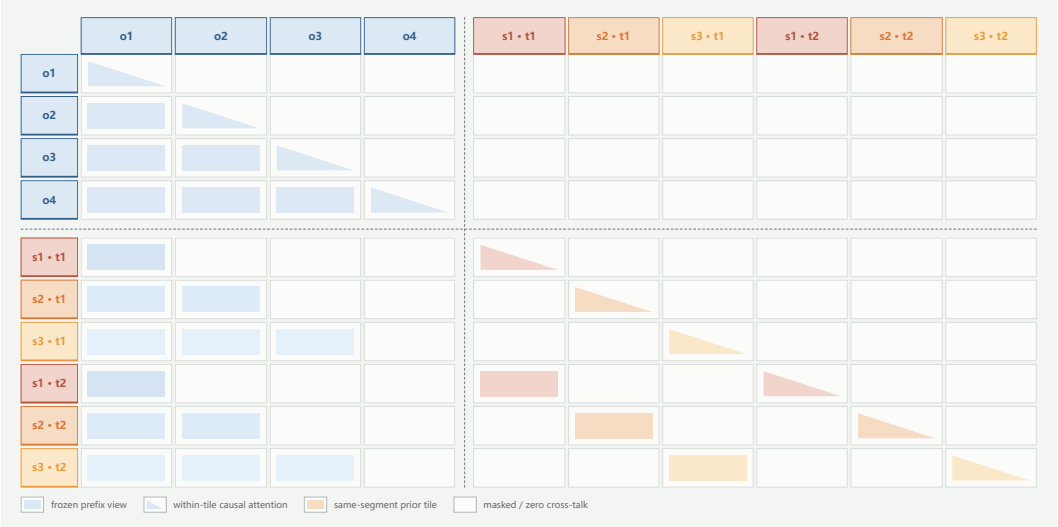


Figure 15: **Attention visibility topology in segment-level teacher-forced rollout (TFR).** Rows represent query blocks and columns represent key blocks across a multi-segment rollout. The frozen canonical prefix block τ^* is computed once in the initial scan pass and remains strictly read-only throughout the episode; prefix queries never reference live rollout slots (top row). Each candidate transition decomposes into execution segments (s_1, s_2, s_3) that advance in parallel wave tiles (t_1, t_2). Queries attend to their anchor-specific prefix views $(\cdot^*)$ and to preceding tiles within the same segment, while enforcing strict causal lower-triangular attention within each tile. Crucially, all off-diagonal segment blocks are completely unreferenced (blank white space), enforcing zero inter-segment crosstalk and guaranteeing that concurrent segment rollouts remain mathematically independent while live tokens write to isolated shadow slots.

inherit the logical coordinates of their target addresses and are visible only to subsequent tokens within the same segment. No generated values overwrite canonical entries or leak into concurrent segment rollouts, ensuring zero inter-segment crosstalk.

The resulting block-visibility topology across segments and waves is illustrated in Figure 15.

Wave-Parallel Horizon and Truncation Accounting. Because segments maintain isolated shadow slots, active segments across an episode advance synchronously token-by-token in parallel waves. The total number of rollout forward passes for an episode is bounded by the length of the longest active segment, $\max_j |g_{t,j}|$, rather than the cumulative token sum of all actions. A segment terminates when emitting any token in its terminator set $\mathcal{T}_j$. If a segment reaches the uniform wave cap $W_{\max}$ before emitting a valid terminator, it is recorded as a truncation rather than an execution failure, preventing artificial bias from rigid token-budget cutoffs.

Hierarchical Semantic Verification. Verification proceeds hierarchically across three explicit levels:

1. **Segment-Level Assessment**: Each generated segment $g_{t,j}$ is evaluated against canonical text $g^*_{t,j}$ (strict match) and local admissible set $\mathcal{L}(S^*_t, j)$ (legal match, accommodating permutations of clause lines).

2. **Transaction Assembly**: Completed segments are composed into a full syntactic transaction candidate $\hat{U}_t = \text{Assemble}(g_{t,1}, \ldots, g_{t,K_t})$. If any constituent segment encountered unexpected termination or was truncated, transaction assembly fails with an informative diagnostic verdict.

3. **Step-Level Transition Validity**: The deterministic environment verifier evaluates whether the assembled transaction induces a valid state transition from Oracle state S_t^*:

$$\mathrm{Valid}(S_t^*, \hat{U}_t) = \begin{cases} 1, & \text{if } \hat{U}_t \text{ parses and } \Phi(S_t^*, \hat{U}_t) \text{ is a valid transition,} \\ 0, & \text{otherwise.} \end{cases} \tag{32}$$

The conditional single-step transition success probability at step t is:

$$p_t = \mathbb{E}_{S_t^* \sim \mathcal{D}_t}\big[\mathbb{P}_{\boldsymbol{g}_t \sim \pi_\theta(\cdot|\mathcal{V}(S_t^*))}\big(\mathrm{Valid}(S_t^*, \mathrm{Assemble}(\boldsymbol{g}_t)) = 1\big)\big]. \tag{33}$$

Oracle-Conditioned Aggregate. The state-anchored protocol deliberately does not estimate free-running trajectory success. A strict or legal zero-error episode records whether every evaluated action on the canonical trajectory is valid under the Oracle reset protocol. For optional diagnostics, one may form the factorized quantity

$$P_{\text{oracle-factorized}} = \prod_{t=0}^{T-1} p_t, \tag{34}$$

but this is a counterfactual product under independent Oracle-state resets, not a measured end-to-end success probability. The reported Stepwise Transition Evaluation therefore uses per-step strict/legal error, failure-free horizon, and zero-error Oracle-episode rate as separate diagnostics.

F THESEUS-AR TRANSITION SEMANTICS

This appendix formalizes the operational semantics, grammar, and execution invariants of the discrete state transition system introduced in Section 3.

F.1 FORMAL SYNTAX OF THE UPDATE LANGUAGE

Let Σ denote the token vocabulary of the underlying language model, and let $\Omega \subset \Sigma$ be the set of reserved special control tokens:

$$\begin{aligned} \Omega = \{&\texttt{<|replace|>}, \texttt{<|with|>}, \texttt{<|commit|>}, \\ &\texttt{<|target_start|>}, \texttt{<|target_end|>}, \texttt{<|to|>}, \\ &\texttt{<|edit_start|>}, \texttt{<|edit_end|>}, \\ &\texttt{<|editable_start|>}, \texttt{<|editable_end|>}\}. \end{aligned}$$

The update language factorizes into an execution target, which identifies the sub-sequence to be excised, and a replacement payload, which emits the new content. In Extended Backus–Naur Form (EBNF), an update U_t is defined by:

$$\begin{aligned} \langle \text{Update} \rangle &::= \texttt{<|replace|>}\ \langle \text{Target} \rangle\ \texttt{<|with|>}\ \langle \text{Replacement} \rangle\ \texttt{<|commit|>} \\ \langle \text{Target} \rangle &::= \langle \text{Exact} \rangle \mid \langle \text{Range} \rangle \\ \langle \text{Exact} \rangle &::= [\langle C_L \rangle]\ [\texttt{<|target_start|>}]\ \langle \text{Old} \rangle\ [\texttt{<|target_end|>}]\ [\langle C_R \rangle] \\ \langle \text{Range} \rangle &::= [\langle C_L \rangle]\ [\texttt{<|target_start|>}]\ \langle A_L \rangle\ \texttt{<|to|>} \\ &\quad\ \langle A_R \rangle\ [\texttt{<|target_end|>}]\ [\langle C_R \rangle] \\ \langle \text{Replacement} \rangle &::= (\Sigma \setminus \Omega)^* \\ \langle \text{Old} \rangle &::= (\Sigma \setminus \Omega)^* \\ \langle C_L \rangle, \langle C_R \rangle &::= (\Sigma \setminus \Omega)^+ \\ \langle A_L \rangle, \langle A_R \rangle &::= (\Sigma \setminus \Omega)^+ \end{aligned}$$

where $\langle C_L \rangle, \langle C_R \rangle$ denote left and right disambiguating context, and $\langle A_L \rangle, \langle A_R \rangle$ denote boundary anchors. Reserved control tokens act as unambiguous syntactic barriers and cannot appear as literal payload within the target or replacement expressions.

Editing Boundaries. An update decision opens with <|edit_start|>, prompted by the runtime environment. The model then predicts an update U_t or emits <|edit_end|> to terminate the transition sequence. In the displayed context, editable regions are enclosed by <|editable_start|> and <|editable_end|>. Disambiguating context tokens outside these markers may be matched by the target resolver, but the excised interval O_t must reside strictly within the permitted editable boundary.

F.2 Deterministic Target Resolution and Invariants

Target resolution is governed by a deterministic mapping $\text{Resolve} : \mathcal{X} \times \langle\text{Target}\rangle \to [j_{\text{start}}, j_{\text{end}}]$, which identifies a contiguous token span $O_t = X_t[j_{\text{start}} : j_{\text{end}}]$ in the active state X_t.

Exact Selection. For an exact selector without <|to|>, the resolution envelope is formed by concatenating the contextual prefixes, the target span, and contextual suffixes:

$$\text{Env}(\langle\text{Exact}\rangle) = \langle C_L\rangle \parallel \langle\text{Old}\rangle \parallel \langle C_R\rangle.$$

The resolver identifies all occurrences of Env in X_t. If both boundary markers <|target_start|> and <|target_end|> are present, only the inner span $\langle\text{Old}\rangle$ is designated as O_t. Omission of <|target_start|> extends the edit start to the left boundary of the envelope; omission of <|target_end|> extends the edit end to the right boundary. Adjacent target markers with an empty $\langle\text{Old}\rangle$ define a zero-width insertion point.

Range Selection. For range targets containing <|to|>, the resolver searches for an ordered, non-overlapping pair of anchors $(\langle A_L\rangle, \langle A_R\rangle)$ conforming to the surrounding contextual constraints. The excised span O_t encompasses both anchors and all intermediate tokens. The interior sequence between anchors is omitted from the predicted target string, enabling constant-length excisions of arbitrarily long spans.

Well-Formedness Invariants. Every valid state transition must satisfy three formal invariants:

1. **Invariant 1 (Unique Resolvability)**: The envelope or anchor pair must resolve to exactly one match in X_t:

   $$\exists!\, [s, e] \subseteq [1, |X_t|] \quad \text{s.t.} \quad X_t[s : e] \models \langle\text{Target}\rangle.$$

   If zero matches or multiple ambiguous matches exist, resolution fails.

2. **Invariant 2 (Strict Region Containment)**: The resolved interval $[s, e]$ must be strictly contained within the bounds of the active editable region: $[s, e] \subseteq [\text{pos}(\text{<|editable_start|>}) + 1, \text{pos}(\text{<|editable_end|>}) - 1]$.

3. **Invariant 3 (State Progression)**: The replacement must effect a non-trivial modification: $X_{t+1} \neq X_t$. Updates with identical replacements are rejected as null operations.

Upon violation of any invariant, the transaction aborts atomically, rolling back uncommitted cache allocations and preserving the previous state S_t.

F.3 Execution Dynamics and Attention Invariants

We use the binary visibility mask $\mathbf{V} \in \{0, 1\}$ in this section, with $V_{q,k} = 1$ for an admissible attention edge and $V_{q,k} = 0$ for a masked edge.

Token-Boundary Expansion. While string matching operates at character resolution, key–value cache manipulation operates at discrete token boundaries. If a character-level target boundary bisects a multi-character token, the execution interval expands to the minimal enclosing token boundary. Border characters outside the intended semantic edit are preserved and automatically prepended or appended to the replacement payload. Committed tokens retain their physical representations across transitions.

Three-Phase Attention Masking and Coordinates. Let $X_t = P_t \parallel O_t \parallel Q_t$. Attention routing enforces:

- **Phase 1 (Target Identification)**: Control tokens through <|with|> attend causally to X_t. The output projection of <|with|> predicts the first replacement token (or <|commit|> if $V_t = \emptyset$).
- **Phase 2 (Replacement Generation)**: For all replacement queries $q \in V_t$, the excised span O_t is masked out of the attention field:

$$\mathbf{V}_{q,k} = 0, \quad \forall k \in O_t.$$

  Replacement queries attend to P_t, Q_t, and generated replacement prefixes. Tokens in V_t are appended physically to the tape, receiving temporary logical coordinates starting at the origin of O_t. If $|V_t| > |O_t|$, suffix positions displace dynamically via a position-expansion policy.
- **Phase 3 (Commit and Coordinate Realignment)**: Upon emitting <|commit|>, the runtime validates invariants and finalizes the transition. The successor state $X_{t+1} = P_t \parallel V_t \parallel Q_t$ is published with contiguous logical coordinates:

$$\ell_{t+1}(j) = \begin{cases} \ell_t(j), & j \in P_t, \\ \ell_t(\text{start}(O_t)) + (j - \text{start}(V_t)), & j \in V_t, \\ \ell_t(j) + (|V_t| - |O_t|), & j \in Q_t. \end{cases}$$

  Representations in O_t and temporary control tokens are retired from the active topology, and <|commit|> is consumed without allocating a physical KV entry.

F.4 Cache Lifecycle and Compaction

During online execution, superseded tokens leave the active sequence view whenever a state transition invalidates them. The physical key–value cache manages storage through two complementary mechanisms:

- **Transaction-Local Tail Commit and Tombstoning**: Upon committing an edit, temporary control tokens are cropped from the physical cache tail, newly materialized entries for V_t (and any refreshed suffix window Q_t^{ref}) remain appended at the physical tail, and superseded entries in O_t (and prior Q_t^{ref}) are masked in place as tombstones ($\mathbf{V}_{q,k} = 0$), producing the uncompacted layout $\mathcal{C}_{t+1}^{\text{raw}} = P_t \parallel O_t^{\dagger} \parallel (Q_t^{\text{ref}})^{\dagger} \parallel Q_t^{\text{keep}} \parallel V_t \parallel Q_t^{\text{ref}}$ without shifting Q_t^{keep} in memory.
- **Threshold-Triggered Logical Compaction**: When accumulated tombstones reduce active slot utilization $|\mathcal{K}_{t+1}|/|\mathcal{C}_{t+1}^{\text{raw}}|$ below a threshold τ_{compact} (default 0.70), an online compaction pass gathers active key–value entries along with their logical coordinates into contiguous logical order $\mathcal{C}_{t+1}^{\text{compact}} = P_t \parallel V_t \parallel Q_t^{\text{ref}} \parallel Q_t^{\text{keep}}$ and reclaims all tombstone holes. Because RoPE rotation uses each entry's co-moved logical coordinate $\ell_{t+1}(j)$ rather than its physical slot index, this rearrangement leaves attention outputs algebraically invariant (Equation equation 29 in Appendix B.5).

F.5 Executable Semantic Traces

The following traces illustrate how the operational grammar resolves exact disambiguation, interval deletion, and zero-width insertion.

Trace 1: Disambiguating an Exact Target. Context contains two identical occurrences of `0`:

```
a=0;b=0;
```

Contextual tokens `a=` and `;` establish unique resolvability (Invariant 1) while restricting the excised span to the first assignment:

```
<|replace|>a=<|target_start|>0<|target_end|>;<|with|>1<|commit|>
```

The resulting successor state is `a=1;b=0;`.

Trace 2: Range Deletion. Given intermediate scaffolding with redundant calculation:

```
pre;begin;x=1;y=2;end;post;
```

Anchored range selection specifies the interval [`begin;`, `end;`] with an empty replacement string:

```
<|replace|>begin;<|to|>end;<|with|><|commit|>
```

The intermediate tokens `x=1;y=2;` are excised without explicit token enumeration, yielding successor `pre;post;`.

Trace 3: Zero-Width Boundary Insertion. Given consecutive statements:

```
a=1;b=2;
```

Adjacent target markers define an empty interval $\langle \text{Old} \rangle = \emptyset$:

```
<|replace|>a=1;<|target_start|><|target_end|>b=2;<|with|>c=3;<|commit|>
```

Suffix `b=2;` is displaced by $\Delta = |V_t|$ tokens (e.g., $+5$ under this illustrative tokenization), producing successor `a=1;c=3;b=2;`.

G Programmatic Testbeds and Data Generation

This appendix provides formal specifications, reduction rules, and generation protocols for the three controlled programmatic tasks: ListOps Reduction, Davis–Putnam Variable Elimination (DP-VE), and Typed λ-Calculus Reduction. Representative token-level samples for all three tasks are collected in Appendix H.

G.1 ListOps Reduction Benchmark

Task Definition. ListOps Reduction evaluates hierarchical expression evaluation on nested prefix operator trees over single-digit integers. The target value is obtained by iterative innermost-first operator folding.

Formal Syntax and Semantics. Let $\mathcal{D} = \{0, 1, \ldots, 9\}$ be the set of scalar digit values. The expression language comprises:

$$\begin{aligned} \text{Value} \quad & v ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\ \text{Op} \quad & \omega ::= \text{MIN} \mid \text{MAX} \mid \text{SUM_MOD} \mid \text{MED} \\ \text{Expr} \quad & e ::= v \mid \omega(e_1, e_2) \mid \text{MED}(e_1, e_2, e_3) \end{aligned}$$

The operators compute the following functions over $\mathcal{D}$:

- $\text{MIN}(a, b) = \min(a, b)$
- $\text{MAX}(a, b) = \max(a, b)$
- $\text{SUM_MOD}(a, b) = (a + b) \bmod 10$
- $\text{MED}(a, b, c) = \text{median}(a, b, c)$ (the median of three digits)

Deterministic Reduction Oracle. A subexpression $\omega(v_1, \ldots, v_k)$ is *ready* if all its operands $v_i \in \mathcal{D}$ are terminal values. Out of multiple potentially ready subexpressions, the canonical oracle always selects the **leftmost innermost ready node** in left-to-right textual prefix order. The transition rule replaces the selected operator node in place with its scalar evaluation result:

$$\ldots \omega(v_1, \ldots, v_k) \ldots \quad \Longrightarrow \quad \ldots v_{\text{result}} \ldots \tag{35}$$

Because exactly one operator node is eliminated per step, the total number of semantic transitions in any valid derivation strictly equals the initial number of operator nodes N.

Non-Degeneracy Guardrails. Instances are generated by top-down constraint propagation from a chosen root answer. The generator rejects absorbing configurations in which a single operand determines the output regardless of the other branches:

- **Absorption**: Configurations where a single operand fixes the output regardless of other branches (e.g., MIN containing 0, MAX containing 9, or MED with two identical arguments).

Identity configurations in which the operator returns one operand unchanged (for example, MAX containing 0 or MIN containing 9) are retained and reported as a diagnostic statistic.

Dataset Parameters. The scale axis uses operator budgets $N \in \{29, 58, 84, 112\}$; because one operator is removed per semantic transition, N is also the trajectory length. Each budget has four structural strata: `balanced`, `left_deep`, `right_deep`, and `random`, with a target ternary operator share of 25% (MED). Each stratum contributes 1,024 training, 12 validation, and 78 test instances, for a total of 4,096/48/312 with structure-disjoint partitions.

Each training episode is stored as one complete update string rather than as a sequence of separately stored states; representative materializations are shown in Appendix H.

G.2 Davis–Putnam Variable Elimination (DP-VE) Benchmark

Task Definition. DP-VE evaluates structured Davis–Putnam variable elimination and unit propagation on propositional logic formulas in conjunctive normal form (CNF). The task requires tracking multi-clause dependencies, resolving complementary literals, and pruning satisfied constraints until reaching a decision verdict (`SAT` or `UNSAT`).

Formal Syntax and State Representation. Let $\mathcal{V} \subseteq \{a, b, \ldots, z\}$ be a set of propositional variables. A literal l is either a variable v or its negation $-v$. A clause $C = \{l_1, \ldots, l_k\}$ is a disjunction of literals, and a formula $F = \{C_1, \ldots, C_m\}$ is a conjunction of clauses. The active workspace state is rendered as:

- **Quantifier Prefix** (`[∃]`): Lists the active uneliminated variables in their scheduled elimination order: `[∃]` v_1 v_2 $\ldots v_k$.
- **Clause Matrix** (`[F]`): Contains the active clauses, rendered one clause per line with space-separated literals. The empty clause is denoted by `[]`.

Priority-Ordered Elimination Rules. At each step t, the deterministic scheduler evaluates the active formula against three mutually exclusive rules in strict priority order:

1. **Rule 1: Unit Propagation (`unit`)**. If a unit clause $C = \{l\}$ exists:
   - Delete all clauses containing literal l (satisfied clauses).
   - Delete literal $-l$ from all remaining clauses containing $-l$.
   - Remove variable $\mathrm{var}(l)$ from the quantifier prefix.
   - If literal deletion produces a clause identical to an existing clause, the duplicate is removed (exact-duplicate normalization).
2. **Rule 2: Variable Elimination (`eliminate`)**. Let v be the leading variable in the quantifier prefix. If clauses containing v exist, select the first such clause C in workspace line order. For every clause D containing $-v$:
   - Form the resolvent $R = (C \setminus \{v\}) \cup (D \setminus \{-v\})$.
   - If R is non-tautological (contains no complementary pair u and $-u$) and not already present in the workspace, append R to the clause matrix.

   Once all resolvents are materialized, delete only the focused clause C from the workspace; subsequent elimination transactions repeat this rule until no clause contains the positive literal of v.
3. **Rule 3: Pure Variable Discharge (`discharge`)**. When no clause in the workspace contains the positive literal v of the leading prefix variable, delete all clauses containing $-v$ and remove v from the quantifier prefix.

Termination Conditions. The derivation halts when one of two terminal conditions is met:

- If the empty clause `[]` is generated (or derived via unit propagation), the formula is unsatisfiable; the model appends `[=] UNSAT`.
- If the clause matrix becomes completely empty, the formula is satisfiable; the model appends `[=] SAT`.

Banded Generation and Fill-In Control. Unconstrained variable elimination can cause the clause count to explode as $|P| \cdot |N|$ (fill-in). The release mixes six structural strata, $(n, m, W) \in \{(8, 32, 8), (8, 30, 8), (9, 30, 9), (9, 33, 9), (16, 30, 8), (14, 32, 8)\}$, with shuffled elimination order, binary-clause share 0.25, and maximum clause width 3. The acceptance band is an edit-count window of 178–199; candidates are then screened by a $\rho = 10$ replay approximation with exact Replay evaluation for the largest instances, under a 4,500-token cap. The 4k release contains 4,000 training, 48 validation, and 500 test instances. All strata retain the peak-clause bound of 200.

G.3 Typed λ-Calculus Reduction Benchmark

The task evaluates deterministic multi-step reduction of typed higher-order functional terms over structured product states. The surface language omits type annotations and internal variable binder IDs, presenting readable identifiers to the model. Types and binding scopes belong to the formal AST and audit artifacts. The core grammar comprises:

$$\tau ::= \alpha \mid \tau_1 \to \tau_2 \mid \tau_1 \times \tau_2$$
$$e ::= x \mid c \mid D_i \mid \lambda x.\, e \mid e_1\ e_2 \mid \langle e_1, e_2 \rangle \mid \mathrm{fst}(e) \mid \mathrm{snd}(e)$$

where α denotes atomic base types, c denotes symbolic constants, and $\langle e_1, e_2 \rangle$ forms a binary pair with corresponding first and second projections.

A problem instance supplies a set of top-level definitions $\{D_i = e_i\}$, declarations of free typed inputs, and a compact initial expression e_{init}. The active sequence state X_t is the textually rendered AST expression at step t. Step $t = 0$ initializes the empty editable answer region with e_{init}. Subsequent steps apply single rewrite rules determined by a deterministic normal-order (leftmost-outermost) traversal until reaching normal form.

Single-Rule Transition Semantics. Each discrete transition $X_t \to X_{t+1}$ applies exactly one rewrite rule to one selected AST occurrence:

- **Definition Expansion (δ):** Replaces a single named reference D_i with its definition body e_i.
- **Beta Reduction (β):** Contracts one application $(\lambda x.\, e_1)\ e_2$ by substituting all free occurrences of x in e_1 with e_2.
- **Capture Avoidance (α):** When a substitution would cause a free variable in e_2 to be captured by an enclosing binder in e_1, the binder and all its bound occurrences are renamed to a fresh lexical identifier. An otherwise selected β-redex requiring capture avoidance emits this necessary α-step first.
- **Pair Projections ($\mathrm{fst}/\mathrm{snd}$):** Evaluates $\mathrm{fst}(\langle e_1, e_2 \rangle) \to e_1$ and $\mathrm{snd}(\langle e_1, e_2 \rangle) \to e_2$.

Higher-Order Program Synthesis. To decouple problem scale from the complexity of active sequence states, the generator builds higher-order transformation programs over a typed binary product state:

1. **Product State Topology:** A balanced or randomized binary pair layout holds W distinct slots (state width). Each slot contains a binary tree of symbolic atoms of depth P (payload depth). In the main release, $P = 1$; $W \in \{3, 4\}$ is used for IID train/validation strata and $W = 5$ for the held-out size strata.
2. **Primitive Transformations:** Two shared primitive functions operate on the product state: (i) a randomized full permutation cycle across slots, and (ii) a random two-slot swap or single-slot copy. Each primitive is a lambda abstraction constructing the output layout from projections of its argument.

3. **Combinator Composition:** Two combinators provide function composition and repeated reuse:

$$\mathrm{Compose} = \lambda f.\, \lambda g.\, \lambda s.\, f\ (g\ s)$$
$$\mathrm{Twice} = \lambda f.\, \lambda s.\, f\ (f\ s)$$

4. **Synthesized Program Construction:** A random syntax tree composed of $\mathrm{Compose}$, Twice, and the primitive transformations builds a deeply nested pipeline applied to a free symbolic input state.

Data Generation and Qualification. The generation pipeline executes three stages to produce controlled instances:

1. **Sampling:** Sample typed state schemas, primitive functions, and combinator expression trees across the configured width, payload-depth, program-leaf, and reuse-depth strata.
2. **Reduction Oracle:** Reduce the term step by step using the normal-order evaluator, logging the AST, rule label, and character-level edit offsets at each step.
3. **Release Qualification:** Solve each candidate with the explicit $10{,}000$-step resource bound, enforce the exact per-stratum quotas, and balance the three result forms on validation and test. The main release does not impose a fixed character or tape-token rejection band; token counts are measured after rendering for downstream training and evaluation qualification.

Representative Trajectory Dynamics. This benchmark exemplifies the non-monotonic structural dynamics of selective discrete recurrence: while Transition Trace must monotonically retain all serialized intermediate expressions in its append-only KV history, Theseus-AR actively re-indexes and reorganizes its KV cache, allowing active context length (W_t) and the visible attention set to contract from peak expansion down to the terminal state. The normal form is a fully reduced, type-correct term. In one third of the instances the program is applied directly, so the normal form is the product state itself; in the other two thirds the normal form wraps that state, either applying a free observer function to it or binding it under an outer λ-abstraction.

H Representative Task Samples

This appendix prints one complete episode of each controlled task as plain text, materialized by the released generators rather than retyped. A sample is the token stream the model reads: the task description and initial state, then one edit envelope per oracle transition, and a terminating <|edit_start|><|edit_end|>. Inside an envelope, <|target_start|>...<|target_end|> delimits the locator that resolves the replaced span (<|to|> marks its anchored-range form) and <|with|> separates the replacement; the commit that applies an edit is an output-only label on the envelope's last token, not an input token.

The text is shared by the in-place realized conditions: Transition Trace and Theseus-AR consume the same oracle edit stream and differ in how it is represented—an append-only history versus a compacted tape with logical RoPE coordinates—so one materialization represents both. The episodes come from minimal generator configurations, because released-scale instances (Appendix G) materialize into far longer sequences.

Listing 1: ListOps Reduction sample: 3 operator nodes, 4 edit transactions, 248 tokens.

```
Reduce the prefix expression to one value.
First copy the starting expression into the empty answer in one
    initialization transaction. Then apply exactly one rule to exactly
    one occurrence per edit transaction: replace the leftmost operator
    node whose operands are all values by that operator's result.
    Continue until the answer is a single value.

Surface syntax:
- value: a single digit 0-9
- operator node: NAME(operand operand ...), operands separated by single
     spaces
```

```
- every operator node carries its own parentheses and no commas are used

Rules:
- MIN(a b): the smaller operand.
- MAX(a b): the larger operand.
- SUM_MOD(a b): the sum of the two operands modulo 10.
- MED(a b c): the middle value when the three operands are ordered.

Starting expression:
MAX(MAX(2 2) MED(0 6 1))

Answer:
<|editable_start|><|editable_end|>
<|edit_start|><|replace|><|target_start|><|target_end|><|editable_end
    |><|with|>MAX(MAX(2 2) MED(0 6 1))<|edit_start|><|replace|><|
    editable_start|>MAX<|target_start|>(MAX(2 2)<|target_end|><|with
    |>(2<|edit_start|><|replace|><|target_start|> MED(0 6 1))<|
    target_end|><|editable_end|><|with|> 1)<|edit_start|><|replace|><|
    target_start|>MAX(2 1)<|target_end|><|with|>2<|edit_start|><|
    edit_end|>
```

Listing 2: DP-VE sample: 4 variables, 5 clauses, 9 edit transactions, 274 tokens.

```
Eliminate every variable of this CNF, then append the verdict.
[∃] lists the remaining variables in elimination order. [F] starts the
    clause list: one clause per line, literals split by single spaces, -
     negates. [] is the empty clause.
Rules, in priority order:
1. unit: a one-literal clause l is assigned; drop every clause
    containing l, erase -l elsewhere, drop its variable.
2. eliminate: resolve the first clause containing the head variable
    against every clause containing its negation, then drop it; repeat
    until no clause contains it.
3. discharge: when no clause contains the head variable, drop clauses
    containing its negation and drop it.
No clause left: append [=] SAT. [] present: append [=] UNSAT.
Formula:
<|editable_start|>[∃] d c a b
[F]
-a b d
-c d
a b c
a -b c
-a b
<|editable_end|>
<|edit_start|><|replace|><|target_start|>-a b d
<|target_end|><|with|><|edit_start|><|replace|><|target_start|>-c d
<|target_end|><|with|><|edit_start|><|replace|><|target_start|> d c<|
    target_end|><|with|> c<|edit_start|><|replace|><|target_start|>a b c
<|target_end|><|with|><|edit_start|><|replace|><|target_start|>a -b c
<|target_end|><|with|><|edit_start|><|replace|><|target_start|> c a<|
    target_end|><|with|> a<|edit_start|><|replace|><|target_start|>-a b
<|target_end|><|with|><|edit_start|><|replace|><|target_start|> a b<|
    target_end|><|with|> b<|edit_start|><|replace|><|target_start|><|
    target_end|><|editable_end|><|with|>[=] SAT
<|edit_start|><|edit_end|>
```

Listing 3: Typed λ-Calculus sample: 3 state slots, 2 program leaves, 23 edit transactions, 1,308 tokens.

```
Normalize the typed lambda-calculus starting expression.
First copy the starting expression into the empty answer in one
    initialization transaction. Then apply exactly one rule to exactly
    one occurrence per edit transaction. Use leftmost-outermost order,
```

```
    continue inside lambda bodies, and keep the current expression
    between the answer markers.

Surface syntax:
- variable: x
- atom constant: "c"
- named definition reference: @D
- lambda: (λx. expression)
- application: (function argument)
- pair: ⟨left, right⟩
- projections: fst(pair) and snd(pair)

Rules:
- delta: replace one @definition by its body.
- beta: apply one lambda and substitute its argument for every
    occurrence owned by that parameter.
- alpha: when needed to avoid capture, rename one binder and all
    occurrences owned by it in one transaction.
- fst: replace fst(⟨left, right⟩) by left.
- snd: replace snd(⟨left, right⟩) by right.

Free inputs:
- (none)

Named definitions:
- @d0 = (λx0. (λx2. (λx3. (x0 (x2 x3)))))
- @d1 = (λx1. ⟨snd(snd(x1)), ⟨fst(x1), fst(snd(x1))⟩⟩)

Starting expression:
(((@d0 @d1) @d1) ⟨"v1", ⟨"v0", "v2"⟩⟩)

Answer:
<|editable_start|><|editable_end|>
<|edit_start|><|replace|><|target_start|><|target_end|><|editable_end
    |><|with|>(((@d0 @d1) @d1) ⟨"v1", ⟨"v0", "v2"⟩⟩)<|edit_start|><|
    replace|><|editable_start|>((<|target_start|>(@d0<|target_end|><|
    with|>((λx0. (λx2. (λx3. (x0 (x2 x3)))))<|edit_start|><|replace|><|
    target_start|>((λ<|to|>))) @d1<|target_end|><|with|>(λx2. (λx3. (@d1
     (x2 x3)))<|edit_start|><|replace|><|target_start|>(λx2. (λx3. (@d1
    (x2 x3)))) @d1<|target_end|><|with|>λx3. (@d1 (@d1 x3))<|edit_start
    |><|replace|><|target_start|>((λx3. (@d1 (@d1 x3))) ⟨"v1", ⟨"v0", "v2
    "⟩⟩<|target_end|><|with|>(@d1 (@d1 ⟨"v1", ⟨"v0", "v2"⟩⟩)<|edit_start
    |><|replace|><|target_start|>(@d1<|target_end|> (@<|with|>((λx1. ⟨
    snd(snd(x1)), ⟨fst(x1), fst(snd(x1))⟩⟩)<|edit_start|><|replace|><|
    target_start|>((λ<|to|>))<|target_end|><|with|>⟨snd(snd((@d1 ⟨"v1",
    ⟨"v0", "v2"⟩⟩))), ⟨fst((@d1 ⟨"v1", ⟨"v0", "v2"⟩⟩)), fst(snd((@d1 ⟨"v1",
     ⟨"v0", "v2"⟩⟩))))⟩<|edit_start|><|replace|>snd(snd((<|target_start|>
    @d1<|target_end|><|with|>(λx1. ⟨snd(snd(x1)), ⟨fst(x1), fst(snd(x1))⟩
    ⟩)<|edit_start|><|replace|><|target_start|>(((λ<|to|>))),<|
    target_end|><|with|>(⟨snd(snd(⟨"v1", ⟨"v0", "v2"⟩⟩)), ⟨fst(⟨"v1", ⟨"v0
    ", "v2"⟩⟩), fst(snd(⟨"v1", ⟨"v0", "v2"⟩⟩))⟩⟩),<|edit_start|><|replace
    |><|target_start|>(snd(⟨snd<|to|>)⟩⟩)),<|target_end|><|with|>(⟨fst(⟨"
    v1", ⟨"v0", "v2"⟩⟩), fst(snd(⟨"v1", ⟨"v0", "v2"⟩⟩))⟩),<|edit_start|><|
    replace|><|target_start|>snd(⟨fst<|to|>)⟩)<|target_end|><|with|>fst(
    snd(⟨"v1", ⟨"v0", "v2"⟩⟩)<|edit_start|><|replace|><|target_start|>(
    snd(⟨"v1", ⟨"v0", "v2"⟩⟩)<|target_end|><|with|>(⟨"v0", "v2"⟩<|
    edit_start|><|replace|><|target_start|>fst(⟨"<|to|>⟩),<|target_end
    |><|with|>"v0",<|edit_start|><|replace|>fst((<|target_start|>@d1<|
    target_end|><|with|>(λx1. ⟨snd(snd(x1)), ⟨fst(x1), fst(snd(x1))⟩⟩)<|
    edit_start|><|replace|><|target_start|>(((λ<|to|>)),<|target_end
    |><|with|>(⟨snd(snd(⟨"v1", ⟨"v0", "v2"⟩⟩)), ⟨fst(⟨"v1", ⟨"v0", "v2"⟩⟩),
     fst(snd(⟨"v1", ⟨"v0", "v2"⟩⟩))⟩⟩),<|edit_start|><|replace|><|
    target_start|>fst(⟨snd<|to|>)⟩)⟩<|target_end|><|with|>snd(snd(⟨"v1",
    ⟨"v0", "v2"⟩⟩)<|edit_start|><|replace|><|target_start|>(snd(⟨"v1", ⟨"
    v0", "v2"⟩⟩)<|target_end|><|with|>(⟨"v0", "v2"⟩<|edit_start|><|
```

```
replace|><|target_start|>snd(⟨"<|to|>⟩),<|target_end|><|with|>"v2",<|
edit_start|><|replace|><|target_start|>@d1<|target_end|> ⟨<|with|>(λ
x1. ⟨snd(snd(x1)), ⟨fst(x1), fst(snd(x1))⟩⟩)<|edit_start|><|replace
|><|target_start|>(((λ<|to|>⟩)))<|target_end|><|with|>(⟨snd(snd(⟨"v1
", ⟨"v0", "v2"⟩⟩)), ⟨fst(⟨"v1", ⟨"v0", "v2"⟩⟩), fst(snd(⟨"v1", ⟨"v0", "
v2"⟩⟩))⟩⟩)<|edit_start|><|replace|><|target_start|>(snd(⟨snd<|to|>)⟩
⟩)<|target_end|><|with|>(⟨fst(⟨"v1", ⟨"v0", "v2"⟩⟩), fst(snd(⟨"v1", ⟨"
v0", "v2"⟩⟩))⟩)<|edit_start|><|replace|><|target_start|>fst(⟨"<|to|>⟩)
)<|target_end|><|with|>"v1", ⟨"v0", "v2"⟩<|edit_start|><|replace|><|
target_start|> fst(⟨"v1", ⟨"v0", "v2"⟩⟩)<|target_end|><|with|> "v1"<|
edit_start|><|edit_end|>
```