UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

DÉPARTEMENT DES SCIENCES ADMINISTRATIVES

WORKING PAPER NO. 31

PRISME: Intervention-Supervised Sparse Workspace Editing for Small Language Models

Design, controlled evaluation, and negative and positive evidence at the 100M-parameter scale

Simon-Pierre Boucher

Département des sciences administratives
Université du Québec en Outaouais

simon-pierre.boucher@uqo.ca

Gatineau – Pavillon Alexandre-Taché
283, boulevard Alexandre-Taché
Gatineau, Québec, Canada J9A 1L8

October 2026

Version 1.0

Abstract

Keywords: small language models; adaptive computation; latent workspace; counterfactual supervision; sparse routing; quality–compute frontier; reproducibility

JEL Classification: C45, C63, C88

1. Related work and novelty positioning

We re-verified every reference at execution time from primary sources (arXiv metadata and full texts, official repositories and licences; Appendix D and `literature/novelty_audit.md`). Four strands are directly relevant.

Adaptive depth and latent recurrence. Universal Transformers (Dehghani et al., 2019) and ACT (Graves, 2016) introduced weight-shared recurrence over depth with learned per-position halting; PonderNet (Banino et al., 2021) made halting probabilistic. Mixture-of-Depths routes tokens around blocks with a fixed capacity (Raposo et al., 2024), LayerSkip trains early exits (Elhoushi et al., 2024), and CALM calibrates confidence-based exits (Schuster et al., 2022). The recurrent-depth line scaled latent recurrence to language model pretraining: Huginn (Geiping et al., 2025), Ouro (Zhu et al., 2025), Mixture-of-Recursions (Bae et al., 2025), looped Transformers (Giannou et al., 2023; Saunshi et al., 2025), and in 2026 RecurTrace (Wang et al., 2026), Elastic Latent Recurrence (Ladhak et al., 2026), Think-at-Hard (Fu et al., 2026; You et al., 2026) and several contemporaries listed in Appendix D. All of these refine the *entire* hidden state of a position (globally), and route depth per token or per sequence.

Slots, workspaces and sparse writes. Recurrent Independent Mechanisms (Goyal et al., 2019) and the Shared Global Workspace (Goyal et al., 2022) update a small number of competing modules per step; Slot Attention (Locatello et al., 2020) and Perceiver / Perceiver AR (Jaegle et al., 2021; Hawthorne et al., 2022) maintain a small bank of latents that cross-attend to a long input; Sparse Access Memory (Rae et al., 2016) writes sparsely into an external memory. Most recently ReFlux (Ye et al., 2026) writes router-selected sparse deep-to-shallow deltas into a frozen backbone, with the backbone unchanged when no route is active. PRISME's workspace differs in being *ephemeral* (reset per token), separate from a read-only cached bank, and edited by discrete 1–2-slot actions rather than by competition among modules.

Loss-gain supervision of routing. The supervision signal we use—the realised change in next-token loss of an executed extra branch—has become standard for *binary* continue/stop decisions in 2025–2026: Ouro's second stage fits its exit gate to observed per-token loss improvements (Zhu et al., 2025); TaH2 calls the same construction "lookahead depth supervision" (You et al., 2026); RecurTrace trains a halting head on an oracle "more depth still helps" label (Wang et al., 2026). Earlier, fitted value iteration and counterfactual risk minimisation established the use of logged propensities for off-policy action evaluation (Ernst et al., 2005; Dudík et al., 2011; Swaminathan and Joachims, 2015). PRISME makes the label *action-conditional* over 24 spatial actions, logs

propensities, and separates gain, cost, uncertainty and harm heads.

Zero-initialised identity insertion. Adding a zero-initialised branch so that a new module starts as the identity is a well-established device (Bachlechner et al., 2020; Alayrac et al., 2022; Zhang et al., 2023); exact bypass of unrouted tokens exists in Mixture-of-Depths and in TaH/TaH2, and ReFlux leaves the frozen backbone unchanged with no active route. The differential readout of Section 3.5 is a modest variant of this device and is *not* claimed as a contribution.

What is and is not new. After this audit, none of the three ingredients of PRISME is new on its own. What has no direct predecessor is their combination: a per-token slot workspace with selected local writes, trained with action-conditional counterfactual gains over a 25-action catalogue with logged propensities, behind an exact zero-budget bypass. Accordingly, the contribution of this paper is empirical and methodological—a controlled test of whether spatial selection supervised by counterfactual gains buys anything over uniform, dense or global alternatives at matched cost—rather than architectural priority. Table 4 in Appendix D gives the claim-by-claim comparison.

2. Formal problem

Autoregressive prediction under a computation budget. Let $\mathcal{V}$ be a vocabulary of size V and $x_{1:T} \in \mathcal{V}^T$ a sequence. A causal language model assigns $p_\theta(x_{t+1} \mid x_{1:t})$ and is trained to minimise the teacher-forced cross-entropy $\mathcal{L}(\theta) = \mathbb{E}\big[-\log p_\theta(x_{t+1} \mid x_{1:t})\big]$. We consider models whose per-position computation is *variable*: for each position t the model may execute a number $n_t \in \{0, \dots, R_{\max}\}$ of additional *actions* before emitting its distribution, each action having a known arithmetic cost. Writing $c(a)$ for the cost of action a (in units of the backbone's own operations per token), the realised cost of position t is $C_t = 1 + \sum_{r \le n_t} c(a_{t,r})$ plus the cost of deciding, and the quantities of interest are the pairs

$$\big(\,\text{quality}(\pi),\ \ \mathbb{E}_t[C_t \mid \pi]\,\big), \qquad \big(\,\text{quality}(\pi),\ \ \text{median}_t\ \text{latency}_t(\pi)\,\big), \tag{1}$$

traced out as the routing policy π and its penalties vary. Quality is either the held-out cross-entropy or, for tasks with programmatic ground truth, the exact-match accuracy of free greedy generation. The research question is whether a policy that chooses *where* to write (which of M latent slots) using supervision from executed counterfactual branches moves the frontier (1) relative to policies that do not choose (uniform or random slot selection, dense updates of all slots) and relative to global latent recurrence at equal cost.

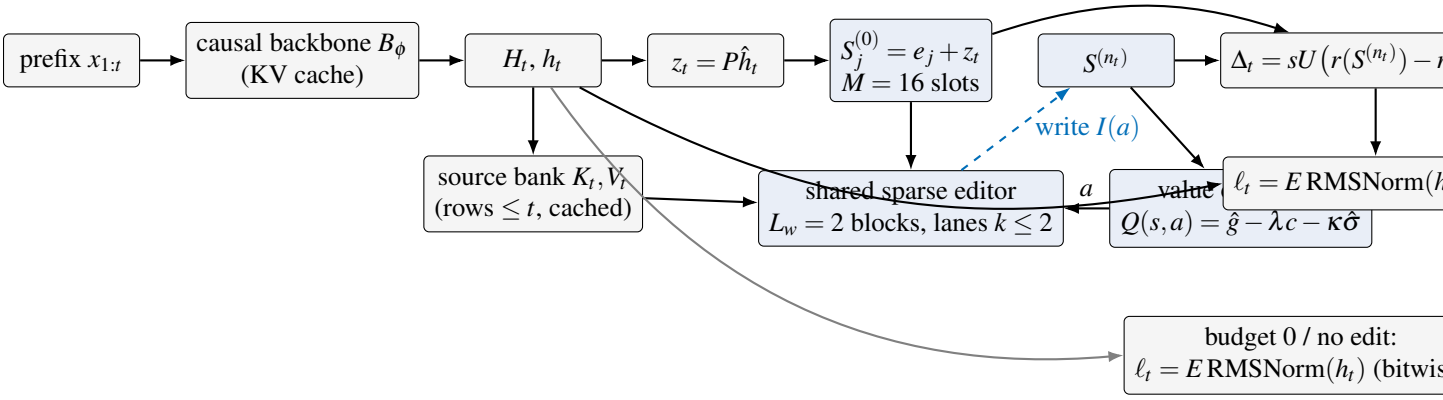


Figure 1. PRISME data flow for one position t. Solid arrows are read paths, the dashed arrow is the only write path (into 1–2 of the M slots per action). The backbone residual stream H, the source bank (K,V) and the summary z_t are inputs to the editor and are never modified by it; the workspace is discarded after the token. With zero executed edits the output is bitwise the backbone output (explicit bypass).

Causal prefix constraint. Every quantity that influences $p_\theta(x_{t+1} \mid x_{1:t})$—including the routing decision—must be a function of $x_{1:t}$ only. In particular the label x_{t+1} is available to the *training* procedure that constructs supervision, but never to the deployed router (Section 4).

What is *not* studied. The model has no persistent memory across positions (the workspace is reset at every t), no retrieval, no tools, no reinforcement learning, and no textual chain of thought. These exclusions are deliberate: they isolate the single mechanism under test.

3. The PRISME architecture

PRISME (Predictive Routing with Intervention-Supervised Memory Editing) adds to a standard decoder a read-only *source bank*, an ephemeral M-slot *workspace*, a shared *sparse editor*, a *value controller* and a *differential readout*. Figure 1 gives the data flow; Table 1 the three dimensioned configurations. All notation is collected in Appendix A.

3.1. Backbone and source bank

The backbone B_ϕ is a pre-norm causal decoder with RMSNorm (Zhang and Sennrich, 2019), rotary position embeddings (Su et al., 2021), grouped-query attention (Ainslie et al., 2023), SwiGLU feed-forward blocks (Shazeer, 2020), tied input/output embeddings and one final normalisation. It returns the final residual stream $H_t = B_\phi(x_{1:t}) \in \mathbb{R}^{t\times d}$, with $h_t = H_t[t]$, and its own head is $\ell_t^{\mathrm{bb}} = E\,\mathrm{RMSNorm}(h_t)$. The *source bank* is a projection of the history to the workspace width

$w \ll d$:

$$\hat{H}_t = \mathrm{RMSNorm}_{\mathrm{bank}}(H_t), \qquad K_t = \hat{H}_t W_K, \qquad V_t = \hat{H}_t W_V, \qquad z_t = P\hat{h}_t, \qquad W_K, W_V, P^\top \in \mathbb{R}^{d\times w}. \tag{2}$$

The bank is distinct from the Transformer KV cache: it stores projected *final* representations, it is append-only, and a query at position t may read rows $1,\dots,t$ only.

3.2. Ephemeral workspace

For every position a fresh workspace $S^{(0)} \in \mathbb{R}^{M\times w}$ is formed from M trainable address embeddings e_j and the current summary,

$$S_j^{(0)} = e_j + z_t, \qquad j = 0,\dots,M-1. \tag{3}$$

e_j carries no imposed semantic role; it only makes slots distinguishable. No state of S survives to position $t+1$; history is carried exclusively by the autoregressive prefix and the two caches.

3.3. Action catalogue

The nonterminal actions are $\textsc{single}_j$ ($j < M$, edit one slot) and $\textsc{pair}_{2i,2i+1}$ ($i < M/2$, edit two fixed slots), plus STOP. For $M = 16$ this gives $|\mathscr{A}| = 24$ nonterminal actions and 25 in total; the $\binom{16}{2} = 120$ arbitrary pairs are *not* available. We write $I(a) \subset \{0,\dots,M-1\}$ for the written index set of action a ($|I(a)| = k(a) \in \{1,2\}$, $I(\textsc{stop}) = \varnothing$). STOP is absorbing and a hard cap $R_{\max} = 8$ bounds the number of actions per position.

3.4. Shared sparse editor

Given the current workspace S, an action a with active set $I = I(a)$, the step index r and the remaining budget $\rho = R_{\max} - r - 1$, the editor forms the active queries $X = S_I + \mathrm{ctx}([r/R_{\max},\ \rho/R_{\max},\ k/2]) \in \mathbb{R}^{k\times w}$ and applies $L_w = 2$ *shared-weight* blocks. Each block computes, with pre-normalisation and residual connections,

$$X \leftarrow X + \mathrm{MHA}\big(q = X,\ k = K_t,\ v = V_t;\ \mathrm{mask}_{\le t}\big) \qquad \text{(read the causal source bank)} \tag{4}$$

$$X \leftarrow X + \mathrm{MHA}\big(q = X,\ k = \bar{S},\ v = \bar{S}\big) \qquad \text{(read all } M \text{ slots of the pre-write snapshot } \bar{S} = S) \tag{5}$$

$$X \leftarrow X + \mathrm{SwiGLU}(X). \tag{6}$$

The bank attention uses query and output projections only (the bank is already in width w and is cached), so the bank is never re-projected per action. The slot attention re-projects the M slot

keys/values at every block (reference implementation; an optimised kernel could cache the keys of unchanged slots). The two lanes of a pair read the *same* snapshot $\bar{S}$, so their writes are simultaneous by construction.

Clipped, gated write. Let $\tilde{S}_j$ be the lane output for $j \in I$. The write is

$$\delta_j = \tilde{S}_j - S_j, \qquad \delta_j^c = \delta_j \min\Big(1, \frac{c}{\max(\mathrm{rms}(\delta_j), \varepsilon)}\Big), \qquad \eta_j = \eta_{\max}\, \sigma\big(g([S_j; \tilde{S}_j])\big), \qquad S'_j = S_j + \eta_j \delta_j^c, \tag{7}$$

and $S'_m = S_m$ *exactly* for $m \notin I$ (the implementation materialises the new rows through a one-hot scatter from the snapshot, so the result is independent of lane order; Proposition 3). We use $\eta_{\max} = 0.25$, $c = 1$: these are untested research hyper-parameters, not tuned values. The gate g and the context map are zero-initialised, so $\eta_j = \eta_{\max}/2$ at initialisation.

3.5. Differential readout and exact bypass

A single attention read with a query that depends on z_t only, $r(S, z_t) = \sum_j \alpha_j S_j$, $\alpha = \mathrm{softmax}_j\big((W_q z_t) \cdot (W_k S_j)/\sqrt{w}\big)$, is evaluated at the final and at the initial workspace, and their difference is mapped back to the residual width:

$$\Delta_t = s\, U \big(r(S^{(n_t)}, z_t) - r(S^{(0)}, z_t)\big), \qquad h_t^{\mathrm{corr}} = h_t + \Delta_t, \qquad \ell_t = E\, \mathrm{RMSNorm}(h_t^{\mathrm{corr}}). \tag{8}$$

$U \in \mathbb{R}^{d \times w}$ is initialised at zero and $s > 0$ is a fixed scalar set once at the start of editor training to the measured root-mean-square of the backbone residual stream (Section 5); because the final RMSNorm is scale invariant, s only reparametrises the learning rate of U. When $n_t = 0$, the term (8) is algebraically zero; the implementation additionally branches out before allocating any slot when the budget is zero, so the fast path performs no workspace computation at all (Proposition 2).

3.6. Configurations and parameter accounting

4. Adaptive intervention policy

The deployed router is a *value controller* that scores every nonterminal action from target-free features and never executes a candidate edit. At state $s = (z_t, S, r, \rho)$ it computes one shared state representation $u = \mathrm{MLP}(W_z z_t + \mathrm{ctx}(r/R_{\max}, \rho/R_{\max}))$ and, for each action a,

$$\bar{s}_a = \frac{1}{k(a)} \sum_{j \in I(a)} S_j, \qquad \big(\hat{g}(s,a), \hat{c}(s,a), \log \hat{\sigma}(s,a), \hat{\ell}_{\mathrm{harm}}(s,a)\big) = W_{\mathrm{out}}\, \mathrm{SiLU}\big(u + W_s \bar{s}_a + \kappa_{k(a)}\big) + b, \tag{9}$$

Table 1. Dimensioned configurations and exact unique parameter counts (meta-device instantiation, `scripts/count_params.py`). Backbone counts equal the reference design exactly; workspace+controller counts differ from the proposal's figures (reconciled in Appendix C).

Property	PRISME-103M	PRISME-459M	PRISME-1.18B
Backbone layers L	12	16	24
Hidden d / FFN	768 / 2048	1536 / 4096	2048 / 5632
Query / KV heads	12 / 3	24 / 6	32 / 8
Workspace w / FFN_w	256 / 768	512 / 1536	768 / 2048
Slots M, blocks L_w, cap $R_{\max}$	16, 2, 8	16, 2, 8	16, 2, 8
Backbone parameters	99,502,848	446,744,064	1,149,339,648
Workspace + controller	2,631,173	10,505,221	21,655,813
Total	**102,134,021**	**457,249,285**	**1,170,995,461**
Proposal's workspace figure	3,088,260	12,336,900	25,779,844

so that scoring all $|\mathscr{A}|$ actions costs one MLP plus $|\mathscr{A}|$ projections of width w (Section 7). The four heads estimate the *intervention gain* $g(s,a)$ of Section 5, a cost proxy, a predictive scale, and the probability that the gain is negative. Actions are chosen greedily,

$$Q(s,a) = \hat{g}(s,a) - \lambda\, c(a) - \kappa\, \hat{\sigma}(s,a), \qquad Q(s,\text{STOP}) = 0, \qquad a^\star = \arg\max_{a \in \mathscr{A} \cup \{\text{STOP}\}} Q(s,a), \quad (10)$$

where $c(a)$ is the *known* analytic cost of the action (ops of one edit plus one controller evaluation, relative to the backbone's ops per token; Section 7); the learned $\hat{c}$ is kept only as a diagnostic regressor. The state is recomputed after every executed edit, and the loop stops when STOP wins or the cap $R_{\max}$ is reached. The penalties (λ, κ) are tuned on calibration rollouts only (Section 8); sweeping λ traces the controller's quality–cost curve.

Known limitations of the rule. The policy is one-step greedy: it cannot represent an action whose value materialises only after a later action. We quantify this myopia directly with a two-action lookahead oracle (Section **??**) rather than assume it away. The rule is also only as good as the calibration of $\hat{g}$; Proposition 5 bounds the one-step loss of the greedy rule in terms of uniform estimation errors but says nothing about multi-step optimality.

5. Learning from interventions

5.1. Paired intervention-value target

For a training state s (a position t with workspace S after r executed actions), an action a and the known next token $y = x_{t+1}$, let $D(\cdot)$ denote the readout-plus-head map of (8) and T_a the editor's

transition. The intervention gain is

$$g(s,a;y) = \mathscr{L}\big(D(s),y\big) - \mathscr{L}\big(D(T_a(s)),y\big), \tag{11}$$

obtained by cloning s, executing a with the same checkpoint, and evaluating both branches on the same y. A positive gain means the edit reduced the teacher-forced loss on this example. This is an intervention on the model's own computation, not a causal effect in the world, and a token-level gain does not imply that a complete generated answer improves (Section **??**). The label y enters (11) only; the runtime features of (9) never contain it (verified by a unit test that perturbs x_{t+1} and checks that all earlier predictions are unchanged).

5.2. Logging policy and propensities

On sampled training states we evaluate four candidates: the current controller's preferred action (or a random single if no controller exists yet), a competitive alternative from the other group (a pair if the preferred action is a single and vice versa), and two uniform draws (one single, one pair). The logging mixture's propensity of each candidate is recorded, enabling inverse-propensity or doubly-robust corrections of selection bias (Dudík et al., 2011; Swaminathan and Joachims, 2015). On an audit subset every one of the 24 actions is executed, giving exact one-step oracle values. Helpful and harmful gains are both kept; sibling branches of one state never cross a partition boundary; every label file records the SHA-256 of the editor checkpoint that produced it, and the controller trainer refuses labels from a different checkpoint.

5.3. Staged optimisation

Stage A (backbone). A common backbone checkpoint is trained (or, for the synthetic experiments, trained from scratch on the synthetic train split). Every comparison starts from the same weights, tokenizer and data.

Stage B (editor bootstrap). The backbone is frozen; the editor, bank projections, slot embeddings and readout are trained with the language loss under *forced* action sequences: each position draws a budget $b \in \{0,1,2,4,8\}$ and b uniformly random actions (singles and pairs balanced), later steps being STOP. Because $U = 0$ blocks all gradient into the editor at initialisation, the bootstrap requires a symmetry-breaking device; we study the zero initialisation, a non-zero initialisation and the scale s of (8) explicitly (Section **??**).

Stage C (counterfactual collection). With the editor fixed, states are sampled by executing $r \sim U\{0,\dots,3\}$ random actions, and labels (11) are collected as above.

Stage D (controller fit). The controller minimises

$$\mathscr{L}_{\text{ctrl}} = \alpha\,\text{Huber}_{\delta}\big(\hat{g}, \text{sg}(g)\big) + \beta\,\text{BCE}\big(\hat{\ell}_{\text{harm}}, \mathbf{1}[g<0]\big) + \gamma\Big(\tfrac{(g-\text{sg}(\hat{g}))^2}{2\hat{\sigma}^2} + \log\hat{\sigma}\Big), \tag{12}$$

on the training partition; sign accuracy, Brier score, expected calibration error and oracle regret are measured on the calibration partition, where (λ, κ) are also selected by actual rollouts.

Stage E (lock). The final checkpoint (editor + controller + selected penalties) is frozen before any locked test split is read. Alternating editor/controller refinement is treated as a later ablation, not a dependency.

No teacher model and no distillation are used anywhere in this paper; the composite loss of the original proposal reduces to the language loss in Stage B and (12) in Stage D.

5.4. Delayed benefits

An action may have negative immediate gain yet enable a later beneficial action. We therefore report, on audit subsets, the exhaustive one-step oracle, the exhaustive two-action oracle ($|\mathscr{A}|(|\mathscr{A}|+1)$ branches per state) and the "greedy first, then best" value, and decompose the gap between the learned policy and the two-step oracle into an *opportunity* term (two-step minus one-step oracle) and a *prediction* term (one-step oracle minus learned policy).

6. Theoretical properties

The statements below are structural properties of the computation graph, not performance guarantees. Proofs are in Appendix B; each is accompanied by a numerical property test in `tests/unit/` that exercises the implemented code on random inputs (tests are evidence about the implementation, not proofs).

Assumption 1 (prefix causality of the components)**.** B_ϕ is causal; the bank visible to a query at position t is $\{(K_j, V_j)\}_{j\le t}$; the controller features at position t are functions of (z_t, S, r, ρ) only.

Proposition 1 (causality)**.** *Under Assumption 1, for every executed action sequence the corrected logits ℓ_t are a function of $x_{1:t}$ only, for all t and for any number of edits $n_t \le R_{\max}$.*

Proposition 2 (identity and exact bypass)**.** *If $n_t = 0$ then $\Delta_t = 0$ exactly and $\ell_t = \ell_t^{\text{bb}}$ in exact arithmetic. In floating point, the implementation returns the backbone head through an explicit branch that performs no workspace computation when the budget is zero, and through the same branch for positions with no executed edit, so the equality is bitwise for identical execution settings. After joint fine-tuning the identity holds relative to the* current *backbone, not to any earlier checkpoint.*

Proposition 3 (exact locality and order independence)**.** *For an action with active set I, $S'_m = S_m$ bitwise for all $m \notin I$, and the result of a pair action does not depend on the order in which its two lanes are processed.*

Proposition 4 (bounded displacement)**.** *With $\mathrm{rms}(v) = \|v\|_2/\sqrt{w}$, after n_j writes into slot j, $\mathrm{rms}\left(S_j^{(R)} - S_j^{(0)}\right) \le n_j\,\eta_{\max}\,c$. The bound is a consequence of clipping and of the triangle inequality; it does not imply convergence of the edits or any decrease of the loss.*

Proposition 5 (one-step selection regret)**.** *Fix a state s and let $J(a) = g(s,a) - \lambda c(a)$ for $a \in \mathscr{A}$ and $J(\mathrm{STOP}) = 0$, with estimates $\hat{J}(a) = \hat{g}(s,a) - \lambda \hat{c}(a)$. If $|\hat{g}(s,a) - g(s,a)| \le \varepsilon$ and $|\hat{c}(a) - c(a)| \le \delta$ for all feasible a, then the greedy choice $\hat{a} = \arg\max \hat{J}$ satisfies $\max_a J(a) - J(\hat{a}) \le 2(\varepsilon + \lambda\delta)$. With the known cost used in this paper, $\delta = 0$. Adding the risk term $-\kappa\hat{\sigma}$ changes the objective and the bound no longer applies as stated.*

Remark 1. None of the propositions bounds the multi-step value of a trajectory; the greedy rule can be arbitrarily sub-optimal on states where the best first action has negative immediate gain. Section **??** measures that opportunity directly.

7. Implementation and compute model

7.1. Reference implementation

The reference implementation is a self-contained PyTorch package (`prisme/`, about 1,100 lines) with the backbone, bank, workspace, editor, controller, readout, caches, counterfactual labelling, oracles and the analytic cost model, plus training, evaluation and baseline packages. In teacher-forced training all T positions of a batch are processed in parallel: the workspace is a tensor of shape $[B,T,M,w]$, the active queries a tensor $[B,T,k,w]$ with $k = 2$ lanes (the second lane is masked for single-slot actions), and the bank cross-attention of (4) is one scaled-dot-product call with a per-query causal mask over the T bank rows. Incremental decoding keeps two caches per request batch—the per-layer Transformer KV cache and the bank (K,V)—and is tested to agree with full-prefix recomputation within 10^{-4} under forced budgets, learned routing, early STOP and mixed edited/unedited positions; the bypass path is bitwise identical to the backbone.

7.2. Analytic operation counts

We count one multiply–add as two operations and ignore normalisations, activations and softmaxes. For a decode step at context length T the backbone costs $2N_{\mathrm{ne}} + 4LTd + 2dV$ operations (N_{ne} the non-embedding parameters; the vocabulary head is included). One editor block with k active lanes

Table 2. Analytic operations per generated token, PRISME-459M, context $T = 4096$ (`scripts/estimate_flops.py`). "Proposal" is the reference table of the original brief; "implemented" is the accounting of Section 7 with the controller evaluated once per executed action plus once to stop. These are arithmetic counts, not measured timings.

Policy	Proposal (10^9)	Implemented (10^9)	editor	controller	readout
Backbone only	1.296	1.296	0.000	0.000	0.000
4 actions, 1 slot each	1.645	1.666	0.256	0.090	0.019
4 actions, 2 slots each	1.767	1.787	0.378	0.090	0.019
8 actions, 2 slots each	2.195	2.219	0.756	0.143	0.019
8 actions, dense 16-slot writes	5.608	5.633	4.169	0.143	0.019

costs

$$F_{\text{block}}(k,T) = \underbrace{4kw^2 + 4kTw}_{\text{bank read}} + \underbrace{4kw^2 + 4Mw^2 + 4kMw}_{\text{slot read}} + \underbrace{6kwf_w}_{\text{FFN}}, \tag{13}$$

which is exactly the arithmetic model of the original proposal; the slot-key refresh $4Mw^2$ dominates for small k and is the obvious target for a cached kernel. One controller evaluation costs $2w^2 + 4w + 6wf_w + |\mathscr{A}|(2w^2 + 8w)$ and the readout $2w^2 + 2(2Mw^2 + 4Mw) + 2wd$. Table 2 reproduces the reference table of the proposal for the 459M configuration at $T = 4096$ and reports the implemented model's accounting; the two agree within 1.3 % (the proposal's controller and readout terms were slightly smaller). The per-action cost proxy used by the controller is $c(a) = \big(L_w F_{\text{block}}(k(a),T) + 8k(a)w + \text{one controller evaluation}\big)/\text{backbone ops}$; at the 103M configuration and $T = 512$ it is 0.080 for a single and 0.100 for a pair, so eight pair edits cost about +83 % of the backbone's operations and dense sixteen-slot writes about +310 %.

7.3. Memory

Parameters at two bytes each: 0.204 GiB (103M), 0.852 GiB (459M), 2.18 GiB (1.18B). Per request at context T: Transformer KV cache $2 \cdot L \cdot T \cdot d_{kv} \cdot 2$ bytes (96 MiB at 459M, $T = 4096$), bank $2 \cdot T \cdot w \cdot 2$ bytes (8 MiB), one workspace $M \cdot w \cdot 2$ bytes (16 KiB). The bank grows linearly with the context: PRISME v1 is not a constant-memory model, and the bank read in (13) is linear in T per active lane. Training stores one workspace per position ($B \cdot T \cdot M \cdot w$ floats) per recurrence step; measured peak memory is reported with every run.

7.4. Latency protocol

Latencies are measured for batch-1 and batch-8 greedy decoding with warm-up, device synchronisation after every token, fp32, a stated prompt length and number of generated tokens, and the cache state recorded (Section **??**). Dynamic routing introduces host-side control flow at every action; this

Table 3. Synthetic suite v1: splits fixed before sampling. n = items after cross-split exact-duplicate removal; "dropped" = duplicates removed. Test splits are locked until pre-registration is frozen.

Split	Families	Templates	Depths	n	dropped
train	arith, bind, fsm, excl, sched	0,1,2	1,2,3,4	23,005	995
calib	arith, bind, fsm, excl, sched	0,1,2	1,2,3,4	1,690	110
dev_iid	arith, bind, fsm, excl, sched	0,1,2	1,2,3,4	3,344	256
dev_template	arith, bind, fsm, excl, sched	3,4	1,2,3,4	2,373	27
dev_depth	arith, bind, fsm, excl, sched	0,1,2	5,6	1,800	0
dev_compose	compose	0,1,2	3,4,5	540	0
test_iid	arith, bind, fsm, excl, sched	0,1,2	1,2,3,4	3,316	284
test_template	arith, bind, fsm, excl, sched	3,4	1,2,3,4	2,328	72
test_depth	arith, bind, fsm, excl, sched	0,1,2	5,6	1,800	0
test_compose	compose	0,1,2	3,4,5	540	0

overhead is part of the measurement and is not removed.

8. Experimental design

8.1. Data

Synthetic compositional suite. Six generator families with programmatic ground truth (Table 3): multi-step arithmetic (arith), variable binding with dependent substitution and a distractor assignment (bind), register updates (fsm), conjunctive constraints with distractor items (excl), rule-based selection under a budget and an exclusion (sched), and a held-out composition of a binding chain feeding an arithmetic expression (compose). Each family has five phrasing templates; training uses templates 0–2 at depths 1–4 and a dedicated seed stream. Evaluation regimes: *iid* (training templates and depths, new seed), *template* (held-out paraphrases 3–4), *depth* (depths 5–6), and *compose* (held-out family). Splits are fixed *before* sampling by (family, template, depth, seed); items whose full text duplicates an item of an earlier split are dropped and counted (995 train, 256 dev-iid, 284 test-iid duplicates; `data/splits/synthetic_v1/manifest.json`). A separate *calibration* stream (seed 3) is used only for controller calibration and penalty selection. Test streams (seed 9) are locked: the evaluation driver refuses to read them until `protocol/locked_test_manifest.json` records the unsealing.

General language corpus. For the 103M experiments we stream the first documents of FINEWEB-EDU (sample-10BT, ODC-By 1.0; Penedo et al., 2024) in canonical shard order, deduplicate exactly, assign documents to train/val/test by content hash (98/1/1 %), and train a byte-level BPE tokenizer (32,768 symbols, digits split individually) on the train split only. The corpus holds 290.0M train,

2.91M validation and 2.88M locked test tokens (`data/processed/fineweb_edu_300m/manifest.json`).

8.2. Systems and controls

All systems in a comparison share the backbone checkpoint, tokenizer, data stream, step count and batch size. Table **??** lists every system with its unique parameter count, training exposure and inference regime. The parameter-matched plain decoder (feed-forward width 2144 instead of 2048, +2.65M parameters) is trained from scratch with the backbone recipe. The dense-slot editor writes all M slots per action with the same weights. The global-recurrence adapter (Section 1) refines the whole top representation with one shared block at width w, parameter-matched to the PRISME workspace, trained with the identical forced-budget recipe and the identical symmetry-breaking device. Reference routers (random, uniform round-robin, fixed pair, entropy threshold, norm-change threshold) reuse the trained PRISME editor and are therefore matched in everything except the choice of action. The exhaustive one-step and two-step oracles read the target and are reported as ceilings only.

8.3. Pre-registration, seeds and uncertainty

The primary outcome, metrics, decision thresholds and seed plan were written to `protocol/preregistration.md` before any test split was read (Appendix E). Differences between systems on the same items use a paired bootstrap over items (10,000 resamples); differences across training seeds use the per-seed statistic with a t-interval. Tokens of one training run are never treated as independent replications of training. Latency is measured on the stated device with warm-up and synchronisation.

References

Ainslie, Joshua, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai, "GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints," in "Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)" 2023.

Alayrac, Jean-Baptiste, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan, "Flamingo: a Visual Language Model for Few-Shot Learning," in "Advances in Neural Information Processing Systems (NeurIPS)" 2022.

Bachlechner, Thomas, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley, "ReZero is All You Need: Fast Convergence at Large Depth," *arXiv preprint arXiv:2003.04887*, 2020.

Bae, Sangmin, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun, "Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-Level Computation," in "Advances in Neural Information Processing Systems (NeurIPS)" 2025.

Banino, Andrea, Jan Balaguer, and Charles Blundell, "PonderNet: Learning to Ponder," in "8th ICML Workshop on Automated Machine Learning (AutoML)" 2021.

Dehghani, Mostafa, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser, "Universal Transformers," in "International Conference on Learning Representations (ICLR)" 2019.

Dudík, Miroslav, John Langford, and Lihong Li, "Doubly Robust Policy Evaluation and Learning," in "International Conference on Machine Learning (ICML)" 2011.

Elhoushi, Mostafa, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A. Aly, Beidi Chen, and Carole-Jean Wu, "LayerSkip: Enabling Early Exit Inference and Self-Speculative Decoding," in "Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)" 2024.

Ernst, Damien, Pierre Geurts, and Louis Wehenkel, "Tree-Based Batch Mode Reinforcement Learning," *Journal of Machine Learning Research*, 2005, *6*, 503–556.

Fu, Tianyu, Yichen You, Zekai Chen, Guohao Dai, Huazhong Yang, and Yu Wang, "Think-at-Hard: Dynamic Looped Transformers for Improved Reasoning," in "Proceedings of the 43rd International Conference on Machine Learning (ICML), PMLR 306" 2026, pp. 32129–32154.

Geiping, Jonas, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein, "Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach," *arXiv preprint arXiv:2502.05171*, 2025.

Giannou, Angeliki, Shashank Rajput, Jy yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos, "Looped Transformers as Programmable Computers," *arXiv preprint arXiv:2301.13196*, 2023.

Goyal, Anirudh, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf, "Recurrent Independent Mechanisms," *arXiv preprint arXiv:1909.10893*, 2019.

_ , Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio, "Coordination Among Neural Modules Through a Shared Global Workspace," in "International Conference on Learning Representations (ICLR)" 2022.

Graves, Alex, "Adaptive Computation Time for Recurrent Neural Networks," *arXiv preprint arXiv:1603.08983*, 2016.

Hawthorne, Curtis, Andrew Jaegle, Cătălina Cangea, Sebastian Borgeaud, Charlie Nash, Mateusz Malinowski, Sander Dieleman, Oriol Vinyals, Matthew Botvinick, Ian Simon, Hannah Sheahan, Neil Zeghidour, Jean-Baptiste Alayrac, João Carreira, and Jesse Engel, "General-purpose, long-context autoregressive modeling with Perceiver AR," in "International Conference on Machine Learning (ICML)" 2022.

Jaegle, Andrew, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira, "Perceiver: General Perception with Iterative Attention," in "International Conference on Machine Learning (ICML)" 2021.

Ladhak, Faisal, Shantanu Acharya, Santiago Akle Serano, Simeng Sun, Boris Ginsburg, and Krishna C. Puvvada, "One Model to Loop Them All: Elastic Latent Recurrence for Language Models," *arXiv preprint arXiv:2610.12582*, 2026.

Locatello, Francesco, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf, "Object-Centric Learning with Slot Attention," in "Advances in Neural Information Processing Systems (NeurIPS)" 2020.

Penedo, Guilherme, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf, "The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale," *arXiv preprint arXiv:2406.17557*, 2024.

Rae, Jack W., Jonathan J. Hunt, Tim Harley, Ivo Danihelka, Andrew Senior, Greg Wayne, Alex Graves, and Timothy P. Lillicrap, "Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes," in "Advances in Neural Information Processing Systems (NIPS)" 2016.

Raposo, David, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro, "Mixture-of-Depths: Dynamically allocating compute in transformer-based language models," *arXiv preprint arXiv:2404.02258*, 2024.

Saunshi, Nikunj, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi, "Reasoning with Latent Thoughts: On the Power of Looped Transformers," in "International Conference on Learning Representations (ICLR)" 2025.

Schuster, Tal, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler, "Confident Adaptive Language Modeling," in "Advances in Neural Information Processing Systems (NeurIPS)" 2022.

Shazeer, Noam, "GLU Variants Improve Transformer," *arXiv preprint arXiv:2002.05202*, 2020.

Su, Jianlin, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu, "RoFormer: Enhanced Transformer with Rotary Position Embedding," *arXiv preprint arXiv:2104.09864*, 2021.

Swaminathan, Adith and Thorsten Joachims, "Counterfactual Risk Minimization: Learning from Logged Bandit Feedback," *arXiv preprint arXiv:1502.02362*, 2015.

Wang, Yuxiang, Kunyu Feng, Yingda Shen, Haoning Xu, Junyu Wang, and Zhizheng Wu, "RecurTrace: Adaptive Latent Reasoning with Loop-Time Memory," *arXiv preprint arXiv:2609.03379*, 2026.

Ye, Wencheng, Anning Hu, Xiangdong Zhang, Tianyi Wang, Yikang Li, Hengyu Jin, Bing Li, and Junchi Yan, "Write Back the Δ: Revisiting the Same Tokens with Fresh Representations," *arXiv preprint arXiv:2609.32457*, 2026.

You, Yichen, Tianyu Fu, Aosong Feng, Xingtai Lv, Xuefei Ning, Ning Ding, and Yu Wang, "Improving Test-Time Scaling with Adaptive Looped Transformers," *arXiv preprint arXiv:2609.35748*, 2026.

Zhang, Biao and Rico Sennrich, "Root Mean Square Layer Normalization," in "Advances in Neural Information Processing Systems (NeurIPS)" 2019.

Zhang, Lvmin, Anyi Rao, and Maneesh Agrawala, "Adding Conditional Control to Text-to-Image Diffusion Models," *arXiv preprint arXiv:2302.05543*, 2023.

Zhu, Rui-Jie, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian, "Scaling Latent Reasoning via Looped Language Models," *arXiv preprint arXiv:2510.25741*, 2025.

Appendices

A. Notation

Symbol	Meaning
$x_{1:t}$, V	token prefix; vocabulary size (32,768)
B_ϕ, H_t, h_t, d	backbone, residual stream, its last row, backbone width
$\hat{H}_t$, K_t, V_t, z_t, w	bank-normalised stream, source-bank keys/values, summary, workspace width
M, e_j, $S^{(r)}$	number of slots (16), address embeddings, workspace after r actions
$\mathscr{A}$, $I(a)$, $k(a)$, STOP, $R_{\max}$	nonterminal actions (24), written index set, group size, stop, cap (8)
L_w, f_w	editor blocks (2, shared), editor FFN width
$\delta_j, \delta_j^c, \eta_j, c, \eta_{\max}$	proposed change, clipped change, gate, clip threshold (1), max step (0.25)
$r(S,z)$, U, s, Δ_t	attention read, readout matrix (zero-init), fixed scale, correction
$g(s,a;y)$, $\hat{g}, \hat{c}, \hat{\sigma}, \hat{\ell}_{\text{harm}}$	intervention gain and its four predicted heads
$Q(s,a)$, λ, κ, $c(a)$	action score, cost and risk penalties, analytic cost of an action
n_t, C_t	executed edits at position t, realised relative cost

B. Proofs

Proof of Proposition 1. By induction on the number of executed actions. Base case: $S_j^{(0)} = e_j + P\,\mathrm{RMSNorm}(h_t)$ with $h_t = B_\phi(x_{1:t})[t]$, a function of $x_{1:t}$ by causality of B_ϕ. Inductive step: if $S^{(r)}$ and the controller features at step r depend on $x_{1:t}$ only, then the action a_r does (it is a function of those features under Assumption 1), the editor's inputs are $S^{(r)}$, the context scalars, and the bank rows $\{(K_j, V_j)\}_{j \le t}$, each a function of $H_t[1{:}t]$ and hence of $x_{1:t}$; so $S^{(r+1)}$ depends on $x_{1:t}$ only. The readout and head are deterministic functions of $(S^{(n_t)}, S^{(0)}, z_t, h_t)$. In the parallel teacher-forced implementation the per-query mask $j \le t$ enforces the same restriction although all of H_T is resident in one tensor; the unit test perturbs $x_{t+1:T}$ and verifies that $\ell_{1:t}$ is unchanged. □

Proof of Proposition 2. If $n_t = 0$ then $S^{(n_t)} = S^{(0)}$ and $r(S^{(n_t)}, z_t) - r(S^{(0)}, z_t) = 0$, hence $\Delta_t = sU0 = 0$ and $h_t^{\mathrm{corr}} = h_t$. In the implementation two separate branches return the backbone head directly: one when the requested budget is zero (before any slot is allocated or any bank row is read), and one for every position whose edit count is zero after routing; both compute exactly the expression $E\,\mathrm{RMSNorm}(h_t)$ on the same tensor h_t and are therefore bitwise equal to the backbone-only forward on the same device with the same kernels (tests 02–04 and the cache test). The statement about joint fine-tuning is immediate: h_t is whatever the current backbone produces. □

Proof of Proposition 3. The new workspace is $S' = \mathbb{1}[m \in I] \odot C + \mathbb{1}[m \notin I] \odot S$ where $C_m = \sum_i \mathbb{1}[\iota_i = m]\, \tilde{S}'_{\iota_i}$ sums the (at most two) lane results $\tilde{S}'$ over the lane indices ι_i of the action. For $m \notin I$ the first term is zero and $S'_m = S_m$ by the identity $0 \cdot x + 1 \cdot y = y$, which is exact in IEEE arithmetic for finite y. For a pair, the two lanes have distinct indices, so C_m is a single lane result plus exact zeros; both lanes read the same snapshot $\bar{S}$ and neither reads the other's output, so permuting the lanes permutes the summands of an exact sum of one nonzero term and zeros, leaving C unchanged. □

Proof of Proposition 4. Each write into slot j adds $\eta_j \delta_j^c$ with $0 < \eta_j \le \eta_{\max}$ and $\mathrm{rms}(\delta_j^c) \le c$ by the clipping rule (if $\mathrm{rms}(\delta_j) \le c$ the factor is 1 and $\mathrm{rms}(\delta_j^c) = \mathrm{rms}(\delta_j) \le c$; otherwise $\mathrm{rms}(\delta_j^c) = c\,\mathrm{rms}(\delta_j)/\mathrm{rms}(\delta_j) = c$ up to the ε guard, which can only shrink the step). Since $\mathrm{rms}(\cdot) = \|\cdot\|_2/\sqrt{w}$ is a norm, the triangle inequality gives $\mathrm{rms}(S_j^{(R)} - S_j^{(0)}) \le \sum_{\text{writes into } j} \eta_j\, \mathrm{rms}(\delta_j^c) \le n_j \eta_{\max} c$. □

Proof of Proposition 5. Let $a^\star = \arg\max_a J(a)$ over $\mathscr{A} \cup \{\textsc{stop}\}$ and $\hat{a} = \arg\max_a \hat{J}(a)$. Both J and $\hat{J}$ assign 0 to STOP. For every $a \in \mathscr{A}$, $|\hat{J}(a) - J(a)| \le |\hat{g} - g| + \lambda |\hat{c} - c| \le \varepsilon + \lambda\delta =: \Delta$, and $|\hat{J}(\textsc{stop}) - J(\textsc{stop})| = 0 \le \Delta$. Then $J(a^\star) - J(\hat{a}) = \big(J(a^\star) - \hat{J}(a^\star)\big) + \big(\hat{J}(a^\star) - \hat{J}(\hat{a})\big) + \big(\hat{J}(\hat{a}) - J(\hat{a})\big) \le \Delta + 0 + \Delta = 2\Delta$, the middle term being non-positive by the choice of $\hat{a}$. □

C. Parameter reconciliation

The backbone counts of Table 1 are reproduced exactly by $N_{\text{bb}} = Vd + L(2d^2 + 2d\,d_{kv} + 3df + 2d) + d$ with $d_{kv} = n_{kv} \cdot 64$ (tied embeddings, two RMSNorm vectors per block, one final norm). The proposal's workspace+controller figures (3,088,260 / 12,336,900 / 25,779,844) are reproduced exactly—for all three configurations—by the closed form

$$N_{\text{ws}}^{\text{prop}} = 4dw + Mw + 17w^2 + 6wf_w + 15.5\,w + 4,$$

which we inferred algebraically from the three published values. The implemented design counts

$$N_{\text{ws}}^{\text{impl}} = d + 3dw + Mw + \underbrace{(6w^2 + 3wf_w + 3w)}_{\text{editor block}} + \underbrace{(4w + 2w + 1)}_{\text{ctx, gate}} + \underbrace{(2w^2 + wd)}_{\text{readout}} + \underbrace{(2w^2 + 3w + w + 3wf_w + 2w + 4w +}_{\text{controller}}$$

i.e. $10w^2$ square projections instead of $17w^2$ and a few extra vectors: the editor reads the cached bank without re-projecting it (no K/V projections, $-2w^2$), the readout uses raw slot values ($-w^2$), and the controller shares one state MLP across actions ($-4w^2$). The difference is 457,087 (103M), 1,831,679 (459M) and 4,124,031 (1.18B) parameters. We did not add inert projections to match the proposal's figures.

D. Literature audit

All 78 references of this paper were resolved through the arXiv Atom API or the publisher page at execution time (2026-10-29/30); mechanisms of the closest neighbours were read from their full texts, and code availability and licences from the official repositories (`literature/papers.bib`, `comparison_matrix.csv`, `novelty_audit.md`). Table 4 summarises the claim-by-claim audit.

Near neighbours published after June 2025 and re-checked at execution time include Ouro, Mixture-of-Recursions, PonderLM, Inner Thinking Transformer, Encode–Think–Decode, Parallel Loop Transformer, Think-at-Hard and TaH2, LoopFormer, AdaPonderLM, latent lookahead, LoopUS, the Latent Recurrent Transformer, LaTER, Depth-Attention, DiscoLoop, Logan, RecurTrace, ReFlux and Elastic Latent Recurrence (all in `papers.bib`); the arXiv listing should be re-checked the week before any submission.

Table 4. Claim-by-claim novelty audit (2026-10-29).

Proposed claim	Closest sources	Actual distinction	Status
C1 selected local writes into a per-token slot workspace	RIMs, Shared Global Workspace, Perceiver AR, Slot Attention, SAM, ReFlux	no prior work combines a non-persistent per-token slot bank, a separate read-only cached source bank, and discrete 1–2-slot write actions	partially novel
C2 action-conditional counterfactual gain supervision with logged propensities	Ouro stage II, TaH2 "lookahead depth supervision", RecurTrace oracle halting; Dudík et al., Swaminathan & Joachims	binary continue/stop versions exist; conditioning the label on which of 24 spatial actions is taken, with propensities and separate gain/cost/risk/harm heads, has no predecessor	partially novel
C3 differential readout with exact zero-budget bypass	ReZero, Flamingo, ControlNet, Mixture-of-Depths, TaH/TaH2, ReFlux, ELR	a modest variant of zero-initialised identity insertion	not novel
C4 the combination C1+C2+C3	ReFlux, TaH2, RecurTrace, MoR	no work has all three; the contribution must be empirical	partially novel
C5 catalogue of 16 singles + 8 fixed pairs + STOP	top-k expert/mechanism routing, DQN discrete action sets	a design choice; reported with sensitivity analysis	not novel

E. Pre-registration and deviations

Pre-registration — WP31 PRISME pilot (DRAFT v0.1, 2026-10-29; to be FROZEN before any `test_*` split is read)

Status: DRAFT. No file named `test*.jsonl` / `test.bin` has been opened by any evaluation script so far (`protocol/locked_test_manifest.json` records the split hashes and the unsealing timestamp once frozen).

1. Primary benchmark suite

- **Synthetic compositional suite v1** (`data/splits/synthetic_v1`, generator `data/generators/synthetic.p` manifest sha256 in `manifest.json`): families arith, bind, fsm, excl, sched (train/dev) + compose (held out). Task unit = one item (prompt + answer). Exclusion: items whose full text duplicates an earlier split (dropped at generation time; counts in manifest). Regimes: `*_iid` (train templates

& depths, new seed), `*_template` (held-out paraphrase templates 3–4), `*_depth` (depths 5–6 > train max 4), `*_compose` (held-out family).
- **General language**: FineWeb-Edu held-out `test.bin` (2.88M tokens, locked; `val.bin` is the development set).
- Independent benchmarks (GSM8K, EvalPlus) are OUT of the primary scope of the 103M pilot (a ~200M-token backbone cannot solve them above chance); they are reported only if a run is executed, as secondary.

2. Primary metrics

- Quality (reasoning): exact-match accuracy of free greedy generation on the synthetic test regimes (pooled `test_iid+test_template+test_depth`, each regime also reported). Quality (language): held-out cross-entropy on `test.bin` at each budget.
- Cost: (primary) measured median end-to-end decode latency per token at batch 1 on the reference device, with the backbone-only latency as the unit; (secondary) analytic ops/token (`prisme/flops.py`).
- Joint superiority: a configuration dominates if quality is higher at $\leq$ the comparator's latency, or latency is lower at non-inferior quality (paired CI excludes zero).

3. Baselines and tuning budgets

Same backbone checkpoint, same data exposure, same mixed-budget training recipe and step count: (a) frozen backbone (budget 0); (b) parameter-matched plain decoder trained from scratch with the backbone recipe; (c) dense 16-slot editor; (d) PRISME editor with random / uniform / fixed-pair routing at budgets 1, 2, 4, 8; (e) entropy and norm-change routers (thresholds tuned on calibration data); (f) global-recurrence adapter (Huginn/ELR-inspired, parameter-matched); (g) learned value controller (λ, κ tuned on calibration rollouts only); (h) exhaustive one-step oracle and two-step oracle (diagnostic only, never plotted as attainable). Hyperparameter budget per adapter: one learning rate (1e-3 for adapters, chosen in Phase 4) and the shared schedule; no per-baseline tuning beyond the router thresholds.

4. Minimum meaningful effect

- Reasoning: +3 percentage points pooled exact match at comparable latency ($\leq$ +10 % of the comparator's), or –20 % median latency at non-inferior accuracy (lower CI bound > –1 pp).
- Language: zero-budget cross-entropy within 1 % (relative) of the frozen backbone (H5 preservation).

- Routing (H2/H3): learned controller loss at matched mean extra-ops within the CI of the best fixed/random schedule is a NULL for H3.

5. Seeds

Editor/controller stage: 5 independent seeds (1–5) at the Phase 4 scale; at 103M as many as the cluster permits within the session (target 3–5, actual count reported). The backbone is a single seed (declared limitation).

6. Uncertainty

Paired bootstrap over task items (10,000 resamples) for accuracy differences within a seed; across seeds, mean $\pm$ SD and a t-interval over seed-level differences. Tokens of a single run are never treated as independent replications of training.

7. Stopping / resource rules

Pilot compute cap: $\leq$ 2 nodes $\times$ 8 h for Stage A, $\leq$ 6 nodes $\times$ 6 h for stages B–D; if a stage cannot complete, the phase is reported as incomplete and downstream claims are withheld.

8. Deviations

Any change after freezing is recorded in `protocol/deviations.md` with justification and timestamp.

F. Hyper-parameters, commands and run index

G. Complete results

H. Failures

FAILURES.md — failed hypotheses, tests and attempts (never deleted)

- **date**: 2026-10-29; **what failed**: `test_incremental_equals_full_prefix_forced`: cached decoding $\neq$ full-prefix forward.; **evidence**: pytest, atol 1e-4; **resolution**: Root cause: forced-action forward let edits resume after STOP; made STOP absorbing (D4).
- **date**: 2026-10-29; **what failed**: rsync exclusion `experiments` also dropped `configs/experiments/`; baseline launch on M3U96b failed twice (`FileNotFoundError`).; **evidence**: `experiments/logs/pretrain_en` (first attempts); **resolution**: anchored pattern `/experiments`.

- **date**: 2026-10-29; **what failed**: Stage B bootstrap degeneracy (brief §13.1 confirmed): with U = 0 and lr 1e-3, 700 steps of forced mixed-budget training left val loss identical at every budget (0.4144 / 0.4143 / 0.4143 / 0.4146 / 0.4164 for budgets 0/1/2/4/8); grad-norm 0.01–0.02. Diagnosis: scale mismatch — backbone residual rms 5.64 vs readout difference rms 0.06, so U must grow by $\times 100$ before edits can affect logits.; **evidence**: `experiments/runs/p4_editor_v1_seed01_*/metrics.jsonl` (stopped at step ~750); **resolution**: D9: fixed non-learned `readout_scale` = measured backbone residual rms at bootstrap (keeps U = 0 init and exact identity; equivalent to a scale-aware learning rate). Variants with nonzero U init (std 0.02) and lr 3e-3 kept running on M2U64 / M4M64c as evidence.
- **date**: 2026-10-30; **what failed**: Editor collapse at lr 3e-3 (w = 128 mixed budgets): after 500 steps every budget gives exactly the budget-0 loss (0.4005) and the trainable gradient norm is 0.00 — the adapter learned to make all edits inert (gate/U $\rightarrow$ 0) instead of useful.; **evidence**: M4M64a `experiments/logs/p4_diag_mixed_w128_lr3e3.log`, run `p4_diag_mixed_w128_lr3e3_seed01_*`; **resolution**: "harmless collapse" is a real failure mode of the bootstrap; lr 1e-3 retained.
- **date**: 2026-10-30; **what failed**: Global-recurrence baseline run crashed at the final exact-match evaluation (`AttributeError: 'GlobalRecurrenceModel' object has no attribute 'W_K'`): the cached generator assumed PRISME caches. Training and the step-3000 checkpoint are intact.; **evidence**: M5M36 `experiments/logs/p4_gr_mixed_w64.log`; **resolution**: added `generate_uncached` fallback (full recomputation) for models without the bank; EM re-run from the checkpoint.
- **date**: 2026-10-30; **what failed**: Joint fine-tuning reference (backbone unfrozen, lr 3e-4, answer-span loss): validation loss rose from 0.4005 to 0.472 at every budget after 500 steps — the backbone degraded under the narrower objective and the large learning rate.; **evidence**: M4M64a `experiments/logs/p4_joint_mixed_w64.log`; **resolution**: Not a usable upper reference as run; a low-lr variant is deferred (not required by the pre-registered comparison).
- **date**: 2026-10-30; **what failed**: v2 tiny backbone (5.6M, 15k planned steps) overfit the 23k-item train split: validation loss (all tokens) 0.246 at step 6k $\rightarrow$ 0.282 at step 9k. Run stopped at step 9,300; the step-5,000 checkpoint (val 0.249) is used as the frozen v2 backbone.; **evidence**: `experiments/logs/p4_tiny_backbone_v2.log`, run `p4_tiny_backbone_v2_seed01_*` (state: stopped); **resolution**: 42 epochs of a small synthetic set is too many; schedule not re-run (the checkpoint is adequate as a frozen comparator).
- **date**: 2026-10-30; **what failed**: Killed to free nodes for the 5-seed matrix: the degraded joint-finetune reference (M4M64a) and the v1 dense-writes run on the slow m4mn node (superseded by the v2 matrix `p4v2_dense`).; **evidence**: manifests remain in `experiments/runs/` on the nodes (state: running $\rightarrow$ killed)