A COUNT, NOT A FRACTION: HUNDREDS OF EXAMPLES TO BUILD A CACHE, DOZENS TO TRUST IT

Anonymous authors
Paper under double-blind review

ABSTRACT

How much data does it take to install a mechanism in a language model? Measurements at pretraining scale say the answer is a number of examples rather than a share of the corpus, and they say it without saying why. This paper reproduces that law in a system small enough to write down every path the information can take. A 20M-parameter decoder is trained on a task whose attention mask keeps the query away from the source it must report. A shortcut to a visible copy of that source is open in most training examples, blocked in a chosen number of them, and open but carrying a wrong value in a chosen number more. We expected a relay. What forms instead is a one-hop cache, written into the only positions that both see the source and are read by the query, while the positions after them decode the answer and cannot move it. In a pilot version of the task, formation is governed by the count of examples that force it rather than by their share of the data, and the transition does not move when the dataset grows by an order of magnitude. A model written after the experiments, in which each route's contribution is a product of a write and a read, reproduces this. An example the shortcut already answers trains both routes in proportion to the attention each receives, so it cannot shift the balance between them, and with gradient clipping the same accounting locates, after the fact, the one delivery schedule that breaks the law, which packs the forcing examples into too few updates. Once the cache exists, a much smaller count of corrupted examples flips which route wins a conflict; across dataset sizes that dose stays a count only when the learning rate decays at the end of training. That flip is a re-weighting applied to every input rather than a detector for the conflicting ones, so most of the answer changes hands while clean accuracy stays where it was.

1 INTRODUCTION

A denial-of-service backdoor planted during pretraining needs about 250 poisoned documents whether the model has 600M or 13B parameters and whether it trains on 6B or 260B tokens, while 100 were not enough (Souly et al., 2025); the authors conjecture that what matters is the number of sequential gradient steps on poisoned data, but give no mechanism. The opposite regime also exists, and we call it the *dilution view*: every example competes with every other for the same weights, so ten times more other data dilutes a fixed handful of special examples tenfold. Knowledge storage behaves this way when useful facts are mixed 1:7 with junk the model cannot learn, although not when the junk is highly repetitive (Allen-Zhu & Li, 2025). Which regime governs the installation of a mechanism, and why would it be a count rather than a fraction?

We answer it in a system whose routes can be enumerated: a 20M-parameter decoder trained from scratch to report a verb stated in an earlier *source*. Its attention mask keeps the answer position, the *query*, from ever reaching the source, but most training examples give the query a shortcut, the *copy route*, to a visible copy of the source (Section 3). Out of N training examples we set two counts: E *blocked* examples, in which the copy route is cut, and C *corrupted* examples, in which it is open but the copy carries a wrong verb; the other $N - E - C$ examples are *clean*. Because loss falls only on the answer tokens and the mask is fixed, an intervention on a named group of positions rules a whole family of routes in or out. The two doses of Figure 1 come from different task versions:

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.

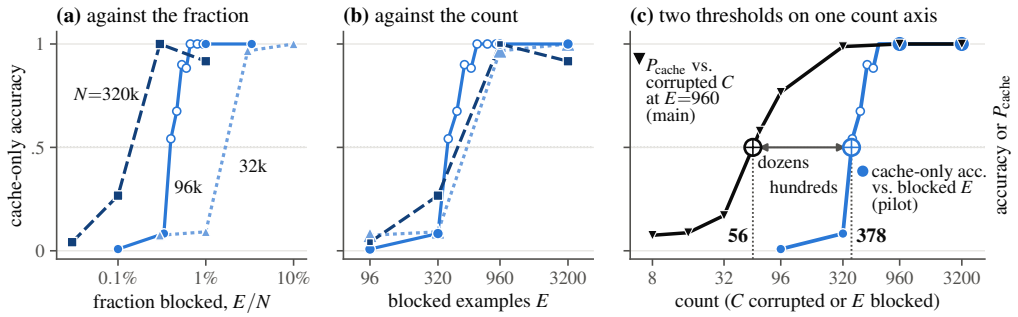


Figure 1. Hundreds to build a cache, dozens to trust it. (a, b) Pilot task. (a) Cache-only accuracy (shortcut blocked at test time) against the fraction of blocked training examples at N=32k, 96k, 320k (light to dark): the transitions sit a decade apart. (b) Against the count E they coincide; hollow: extra doses at 96k. (c) Two task versions on one count axis: cache-only accuracy against blocked examples (blue; pilot task, 96k) and, with the cache built from E=960, its conflict win rate P_{cache} (items whose copy shows a wrong verb) against corrupted examples C (black; main task, 96k, two item sets pooled). Rings: half-crossings, 378 (linear between grid doses) and 56 (log-linear).

the formation dose from a pilot whose shortcut is a direct attention edge to the source rather than a visible copy (Section 4.3), and the switch dose (which route wins a conflict) from the main task.

Three questions follow: where the answer is kept, how many examples build that store, and which route wins a conflict. For each, our guess before the runs was wrong. We expected a *relay*, the verb passed from one filler position to the next until the query reads it; instead it is written once into a *cache*. Between the source and the query lie *filler* positions, loss-free placeholder tokens; the cache is the twelve of them (30–41) that both see the source and are read by the query: token positions, not a key-value cache. The cache is built by a count of blocked examples, not the fraction the dilution view predicts. A conflict is settled by re-weighting both routes on every input, not by detecting conflicts.

We give a model of why both doses are counts. Attention makes each route's contribution a product of a write (how strongly the verb is stored in the route) and a read (how strongly the query attends to it); on a log scale, a training example changes such a product in proportion to the error the model still makes on it, whatever the product's size. A clean example is answered by both routes, so its remaining error trains both, each in proportion to the attention the query gives it; a quantity we call their *balance*, I (Section 4.2), therefore does not change. Only blocked examples, which the copy cannot reach, and corrupted ones, which it gets wrong, move the balance. Formation must carry the cache from near zero and takes hundreds of blocked examples, whereas the switch only has to tip the query's attention and takes dozens of corrupted ones (Figure 1c). The dataset size barely enters: as long as the cache stays small while the copy is trained, the clean examples' total contribution to the cache is bounded independently of their number, because the cache's share of attention shrinks as the copy grows (Section 4.2). The model reproduces a formation dose that does not move with N, but puts the switch at about twice the measured dose (Section 5.3). We wrote it after the experiments; the runs of Section 5.3 test P9, which an earlier version of the model already predicted.

Contributions.

1. **Where the answer is kept.** We locate the answer with interventions: it is a distributed value code in the positions that see the source and feed the query; those blind to it decode it but cannot move it (Section 4.1).
2. **How many examples build it.** We show that in the pilot task a count, not a fraction, sets formation: the half-crossing lies in $(320, 960]$ blocked examples at 32k and 320k and is interpolated at 378 at 96k, though the fraction changes tenfold. At a fixed count, four of five schedules form it (front-loading roughly halves its margin); eight dense bursts fail, which a gradient-clipping bound accounts for if a burst pushes the gradient norm several times past the clipping threshold (Section 4.3).
3. **How a conflict is settled.** We show that once the cache exists, dozens of corrupted examples of 96,000 flip a conflict's winner (pooled half-crossing 56; bracket $(32, 64]$ on one set of test items, $(32, 96]$ on a second). Across two dataset sizes the dose stays below the band a fraction

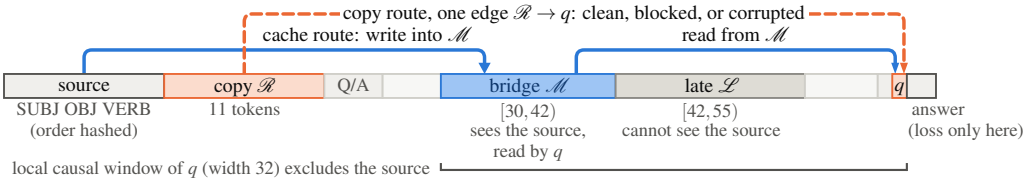


Figure 2. The instrument. A source states three fields in a hashed order; a retrieval copy $\mathscr{R}$ follows, then filler; the query q asks for the verb, and only the answer tokens carry loss. With a local causal window of 32, q never reaches the source. The copy route is one edge from q to $\mathscr{R}$; the cache route runs through the bridge positions $\mathscr{M}$, the only ones that both see the source and are seen by q. The late positions $\mathscr{L}$ cannot see the source.

predicts at the late-window readout under both learning-rate schedules. It falls inside the count band only under the decaying rate. The cache's share of the answer logit moves alike on clean and conflicting inputs: the switch re-weights everywhere (Section 5).

2 Related work

Pre-caching and filler positions. Wu et al. (2024) separate positions that compute ahead for a later token (pre-caching) from positions useful only as a by-product of their own prediction (breadcrumbs) by cutting the gradient path; our filler positions carry no loss, so whether they compute ahead is set by the training mixture, not by a gradient cut. Filler positions are known to hold transplantable computation (Pfau et al., 2024; Darcet et al., 2024; Brauer et al., 2026). *Decodability and use.* That a probe can read information from a position does not show that the model uses it; the dissociation goes back to Elazar et al. (2021) and the control-task literature (Hewitt & Liang, 2019; Ravichander et al., 2021; Lepori et al., 2024); Proposition 4.1 states it for this architecture.

Three literatures, one uncontrolled variable each. Poisoning studies state the dose as a count, near-constant across a twentyfold range of clean data, but give no mechanism (Souly et al., 2025). Earlier poisoning studies state the dose as a share of the data, 0.001–0.1% of pretraining tokens (Zhang et al., 2025) or 0.5–2% of a fine-tuning set (Bowen et al., 2025), but hold the dataset size fixed, so they cannot separate a count from a fraction. Knowledge-conflict studies read out reliance on context in pretrained models whose competing mechanisms formed under an uncontrolled distribution (Xie et al., 2024; Du et al., 2024; Kortukov et al., 2024). In-context against in-weights studies set the distribution as a probability or a property (Chan et al., 2022; Ku et al., 2026), and strategies race and share sub-circuits as they are learned (Edelman et al., 2024; Nguyen & Reddy, 2025; Singh et al., 2025; Park et al., 2025; Kawata et al., 2025). Where a mechanism composes weights, its parts are learned in order (Bietti et al., 2023), and a nested product of attention weights gives a slow phase followed by an abrupt drop in loss (Reddy, 2024); our two routes are products of this kind. We differ from all three: we set both doses as exact counts, measure formation and arbitration causally in the same models, and test whether the learned preference depends on a conflict being present.

3 The instrument: task, routes and readouts

3.1 Sequences, the attention graph, and the two routes

An example is a token sequence of length $n = 64$ with an additive attention mask $B \in (\mathbb{R} \cup \{-\infty\})^{n \times n}$ shared by every head and layer (Figure 2). The *attention graph* of B is the directed graph $G(B)$ on $[n]$ with an edge $k \to q$ exactly when $B_{q,k} > -\infty$; we say q *sees* k and write $\mathscr{N}^-(q) = \{k : k \to q\}$. The mask is local causal with window 32. The retrieval copy $\mathscr{R}$ (positions 11–21) repeats the source. Three changes to the local mask make $\mathscr{R}$ reachable only by the answer query $q = 61$: no other position outside $\mathscr{R}$ sees $\mathscr{R}$; a position inside $\mathscr{R}$ sees only $\mathscr{R}$ up to itself; and the edge $\mathscr{R} \to q$ has additive bias 0 when the copy route is open and is removed when it is blocked (Appendix C).

Definition 3.1 (*Spans*). Let $\mathscr{S} = [0, 11)$ be the source, $\mathscr{R} = [11, 22)$ the retrieval copy, $\mathscr{F} = [23, 55)$ the filler, and $q = 61$ the query. Put $\mathscr{M} = \{p \in \mathscr{F} : \mathscr{N}^-(p) \cap \mathscr{S} \neq \emptyset,\ p \in \mathscr{N}^-(q)\}$ and $\mathscr{L} = \{p \in \mathscr{F} : \mathscr{N}^-(p) \cap \mathscr{S} = \emptyset\}$.

Under the mask, the *bridge positions* $\mathscr{M} = [30, 42)$ are the twelve that can see the source and be read by the query, and the *late positions* $\mathscr{L} = [42, 55)$ the thirteen that cannot see the source. A *route* is a directed path in $G(B)$ that ends at q and starts in the source $\mathscr{S}$ or in the retrieval copy $\mathscr{R}$, which holds a token-level copy of the source: a *copy route* starts in $\mathscr{R}$, and a *cache route* starts in $\mathscr{S}$ and ends with an edge $p \to q$, $p \in \mathscr{M}$.

LEMMA 3.2 (*Route decomposition*). *No path enters $\mathscr{R}$ from outside it, and every path from $\mathscr{R}$ to q ends with the edge $\mathscr{R} \to q$, present exactly when the copy route is open. Every path from $\mathscr{S}$ to q enters q from $\mathscr{M}$ or from $[42, 62)$; a path that avoids $\mathscr{M}$ must pass through $[22, 30)$ and need not pass through $\mathscr{L}$. In a blocked example the logits at q therefore depend on the source only through the key and value states at $\mathscr{M} \cup [42, 62)$.*

An intervention on $\mathscr{M}$ acts on every route through it. A donor patch on $\mathscr{L}$ (Section 3.2) that changes nothing rules out relays through $\mathscr{L}$, not those through 55–60 (e.g. $3 \to 29 \to 60 \to 61$), which interventions on $\mathscr{M}$ rule out.

3.2 THE MIXTURE AND THE READOUTS

DEFINITION 3.3 (*The counted mixture, its schedule, and the readouts*). Fix N and counts $E, C \geq 0$ with $E + C \leq N$. A *training stream* is a sequence of N examples in which E are *blocked* (the edge $\mathscr{R} \to q$ is absent), C are *corrupted* (the edge is present and $\mathscr{R}$ carries a wrong verb) and the rest are *clean*; the target is the source's verb in all three, and a *schedule* assigns the blocked and corrupted examples to positions in the stream. A *readout* evaluates a trained model on held-out items with the copy route in one of three states: the *clean* readout opens it with the source's verb A, the *cache-only* readout blocks it, and the *conflict* readout opens it with a different verb B; clean and cache-only accuracy are the rates of answering A, and the conflict win rate P_{cache} is the rate at which A, the cache's answer, wins. The fraction E/N is derived and never set independently.

The main task uses $N = 96{,}000$ training examples, and 32,000 or 320,000 where the dataset size varies. We call one choice of (E, C, N) and schedule a *setting*; each setting is trained from five seeds, and all runs of a seed share one initialization and one master example order (Table 7, Appendix C). On a span of positions we run three tests. Replacing its key and value states by their mean tests whether the span is *necessary*; patching in the states of a donor example with another verb tests whether it is *sufficient*; and a linear decoder tests whether the verb is *available* there.

Model and training. The model is a five-layer, width-512, eight-head decoder with RoPE, RMSNorm and SwiGLU, about 20M parameters, trained with AdamW at batch 128. Test items come in *panels*: a panel is a fixed set of 12 held-out (subject, object) pairs, and a second, disjoint panel was added for the switch (Appendix C). A *half-crossing* is the dose at which the five-seed mean crosses one half, interpolated between the two grid doses around it; a *bracket* $(a, b]$ names the grid doses on either side of the crossing (Appendix H). Seeds are the replicate, so contrasts are seed-paired. Intervals are 1000-replicate percentile bootstraps over seeds, which undercover at five seeds; Appendix H gives Student-t cross-checks, and the text notes where they disagree. We ran seven *stages* (Table 8, Appendix C): four vary the form of the competing route (Section 4.3), and three test the switch across dataset sizes (Section 5.3). Table 2 (Appendix A) collects the notation.

4 WHERE THE CACHE IS, AND HOW MANY EXAMPLES BUILD IT

4.1 THE CACHE IS ONE HOP FROM THE SOURCE, NOT A RELAY

We expected a relay (Section 1). Lemma 3.2 turns that expectation and its rivals into hypotheses that single interventions decide; Appendices D–F report the measurements behind Sections 4 and 5.

With the copy route blocked, a route that avoids the bridge positions $\mathscr{M}$ (30–41) must end with an edge from $[42, 62)$ (Lemma 3.2); a relay through the late positions $\mathscr{L}$ (42–54) predicts that a donor patch there, from an example carrying another verb, moves the answer. This patch changes no answer in any model with a formed cache, and at most 4.2% of test items in any setting. The same patch on $\mathscr{M}$ alone moves the answer as much as on both spans, and mean-ablating $\mathscr{M}$ alone drives P_{cache} to 0.00: routes through $\mathscr{L}$ carry nothing, and no route avoiding $\mathscr{M}$ decides the answer. Yet a linear

decoder reads the verb out of $\mathscr{L}$ at 0.978 or above in those same models. Nor do $\mathscr{M}$'s keys merely tell q where to look: under conflict, mean-ablating $\mathscr{M}$'s keys leaves P_{cache} at 0.92, but ablating its values leaves 0.00, so the verb is carried in the values. Finally, no single position is necessary: P_{cache} stays at 0.86 or above after ablating any one. Nor is any single position sufficient, since a donor patch on one flips at most 0.21 of test items. The whole span is both (0.00 and 1.00), so the code is distributed, and the last two layers read it: ablating $\mathscr{M}$ at layer 3 or 4 alone leaves P_{cache} at 0.76 or 0.54, at any earlier layer at 0.98 or above (Figures 9 and 10, Appendix E).

PROPOSITION 4.1 (*Available but unused*). *Within this architecture there are settings under which the decoder used throughout (ridge regression on the key and value states averaged over a span) reads the verb from $\mathscr{M}$ perfectly, the copy answers every clean item, and neither mean-ablating nor donor-patching $\mathscr{M}$ changes any answer.*

The construction (Appendix B.3) uses uniform attention and routes the answer through the copy alone. For the paper's ridge decoder the converse also holds, up to float32 rounding; for an arbitrary linear classifier it does not, since an intercept reads a coordinatewise XOR code above chance. Hence we date the cache by when interventions start to change the answer, not by when a probe first exceeds one half. A decoder trained on $\mathscr{M}$ transfers to held-out field orders and shifted filler (Appendix E): $\mathscr{M}$ holds the verb itself, though it carries no loss.

4.2 A TWO-ROUTE MODEL OF THE ANSWER LOGIT

Why does building the cache take a count of blocked examples rather than a fraction? We wrote the model below after the stages had run. *The idea:* the query reads the retrieval copy $\mathscr{R}$ and the bridge positions $\mathscr{M}$ through one softmax. A clean example, which the copy already answers, sends its leftover error back along both routes in proportion to the attention each receives, so it cannot move the balance between them; only a blocked example, where the copy is absent, or a corrupted one, where it is wrong, can.

ASSUMPTION 4.2 (*Two routes, one share*). **(A1, two products.)** At the answer position the logit of verb a is

$$z_a = \underbrace{(1-\pi)\,\alpha\, c_a}_{\text{copy}} + \underbrace{\pi\,\gamma\, m_a}_{\text{cache}} + \varepsilon_a, \qquad \alpha = ab, \quad \gamma = w\,r, \tag{1}$$

where c and m are one-hot on the verbs the copy and the cache assert and ε does not depend on the verb. Each route's gain is a product: w is how strongly the verb is written into $\mathscr{M}$'s values, along the direction the query maps to an answer, and r is how strongly the query reads $\mathscr{M}$; a and b are the same two factors for $\mathscr{R}$. That the verb sits in the values is the key/value ablation of Section 4.1. **(A2, a learned share.)** $\pi = \sigma(\theta)$ is the query's share of attention on $\mathscr{M}$ against $\mathscr{R}$, and θ is trained by the same loss; on a blocked example $\mathscr{R}$ is masked and $\pi = 1$. The measured share moves with the corrupted count, and the model's θ follows it (Section 5.2, Table 4). **(A3, one example, one step.)** Each example is one plain gradient step of size η on the cross-entropy over V verbs; training used AdamW at batch 128 with clipping (Appendix B.9).

Gradient flow keeps balanced factors balanced (Du et al., 2018), so we work in the log-gains $X = \frac{1}{2}\log\gamma$ and $Y = \frac{1}{2}\log\alpha$, the coordinates in which Saxe et al. (2014) solve the two-layer case. In them an example moves the log-gain of a route that asserts the right answer by η times that route's share times the residual $1 - p_A$, whatever the gain's size: a small gain, once pushed, grows geometrically (a plateau, then an abrupt exit), and an example the model already answers barely moves anything (Pezeshki et al., 2021). To first order in η (with $s(\gamma) = (V-1)/(V-1+e^{\gamma})$ the residual on a blocked example, B the copy's verb on a corrupted one, $\kappa = 1 - p_A + p_B$):

	ΔX	ΔY	ΔI
clean	$\eta\,\pi(1-p_A)$	$\eta(1-\pi)(1-p_A)$	0
blocked	$\eta\, s(\gamma)$	0	$\eta\, s(\gamma)$
corrupted	$\eta\,\pi(1-p_A)$	$-\eta(1-\pi)p_B$	$\eta\,\pi\kappa$

(2)

A clean example pays the cache π and the copy $1-\pi$ out of the same residual. The *balance*

$$I = X - \int \pi(1-\pi)^{-1}\, dY \tag{3}$$

subtracts from the cache's log-gain what clean examples give it at its current share; it accumulates along the training path and is not a function of the state (Appendix B.5). Two thresholds decide

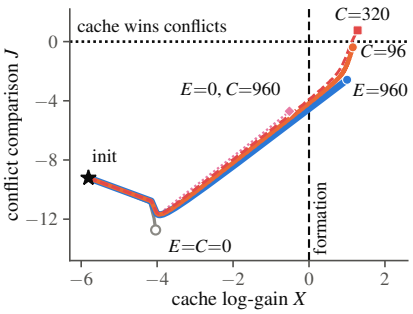


Figure 3. Two thresholds, in the two-route model with a learned share and constants set once, not fitted (η=0.01, V=32, N=96k); model quantities, not measurements. Horizontal: the cache's log-gain X, with the formation threshold dashed ($\gamma = 1$, where a blocked item's answer reaches probability 0.08; probability one half lies further right, and the 960-example path passes it too). Vertical: $J = \theta + \log(\gamma/\alpha)$, positive exactly when the cache wins a conflict (dotted). Clean examples alone raise X by a bounded amount and lower J (grey); with 960 blocked examples the cache crosses the formation threshold the long way (blue); 96 or 320 corrupted examples added to them carry J the short way, to just below and across zero (orange, red). 960 corrupted examples and no blocked ones stop short of formation (pink).

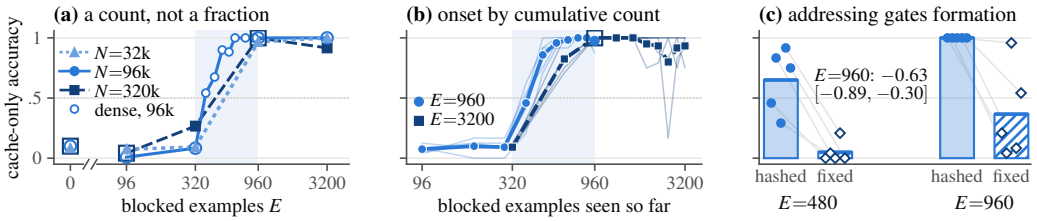


Figure 4. Formation is counted in blocked examples. (a, b) Pilot task. (a) Cache-only accuracy against E at N=32k, 96k, 320k (light to dark; hollow, "dense": extra doses at 96k). (b) Retrainings at 320k read every 32k examples, against the cumulative blocked count (means bold); shaded: 320 to 960, which holds the half-crossing at every N. (c) Main task: copy fields hashed (filled) or at a fixed offset (hatched); bars: means, dots: seeds.

the outcome: the cache answers a blocked item once X reaches a level $X^\star$, and it wins a conflict once $\pi\gamma > (1-\pi)\alpha$, that is once $J = \theta + \log(\gamma/\alpha) > 0$.

THEOREM 4.3 (*One balance, two counts; informal*). *Under Assumption 4.2, to first order in the step: (i) clean examples leave the balance unchanged, so only blocked and corrupted examples move it; (ii) as long as the cache stays small while the copy is trained, the clean examples' total contribution to the cache's log-gain is bounded whatever their number, because the cache's share of attention shrinks as the copy grows, so the number of blocked examples needed to form the cache lies in a range that does not depend on N; (iii) the switch is a second threshold, which corrupted examples approach by raising the cache's share and gain and lowering the copy's, while clean examples push toward whichever route leads in both gain and share. Theorem B.9 states this formally.*

With the unfitted constants of Figure 3 the model needs between 474 and 1285 blocked examples before a blocked item is answered with probability one half, at every N (simulated: 503 at 96k), and it flips the conflict winner at 113 corrupted examples, about twice the measured dose (Section 5.3); three quarters of that change in J comes from the share (Appendix B.7). The hypothesis of (ii) holds in simulation for every schedule the stages ran and fails when all blocked examples come first.

The competitor's form matters because, with unbalanced factors, the formation dose depends on the initialization through $-\log(w_0 + r_0)$. If a competitor searched by content trains the read that q needs for $\mathscr{M}$ (a hypothesis outside the model), a fixed-offset one starts that read lower and needs more blocked examples (Equation 5, Appendix B.8); within a fixed budget only the seeds with the largest initial product then form: unreliable, not impossible (compare Kawata et al., 2025; Singh et al., 2025).

What the model accounts for after the fact (measured before it was written)

P1 (a count, not a fraction). The transition sits at the same count as N varies, and use appears once the *cumulative* count reaches it. **P2 (order matters little).** Any schedule that trains the copy before the cache is large and gives each blocked example its own update builds the cache; a repeat counts again. **P3 (one shape, only shifted).** Seeds inside the transition spread along it rather than split. **P4 (addressing sets reliability).** A competitor read at a fixed offset leaves formation possible but seed-dependent.

4.3 What the formation stages show

Formation is counted in blocked examples (Figure 4), and the count is resolved in the pilot task only: the main task's runs across N use only E=960, far above threshold, so they show that the cache forms, not where (Section 5.3). Table 9 (Appendix D) gives the per-dose and per-seed values behind this subsection. **P1.** At N=96k, cache-only accuracy rises from 0.08 at 320 blocked examples to 0.54 at 384, which puts the half-crossing at about 378. At 32,000 and 320,000 the grid only brackets the crossing, but in the same interval, 320 to 960, although the blocked fraction differs tenfold (a); in retrainings read every 32,000 examples, use arrives once the cumulative count reaches that interval (b). A *gradient probe* takes the inner product of one kind of example's loss gradient with the gradient of a readout margin at a saved checkpoint (Appendix G); the *cache-only margin* is the right verb's logit minus the largest other verb's with the copy route blocked. Before formation, gradient probes put the total push of all clean examples on the cache-only margin between -4.1% and $+0.56\%$ of the blocked examples' push (Table 9).

P2. Repetition counts: 480 distinct blocked examples shown twice build the cache as 960 shown once do. While each blocked example gets its own update, the cache forms whether they are loaded at the front or held to the end, although front-loading roughly halves the cache-only margin. Held to the end, $\mathscr{M}$ already decodes the verb above its untrained floor before any blocked example arrives, yet use waits until 561 have been seen. **P3.** Seeds inside the transition are spread along it rather than split, the long tail Nguyen & Reddy (2025) predict. **P4.** With the copy's fields ordered by a hash of the subject and object, the cache forms in all five seeds; at a fixed offset, with mask, tokens, positions and initial weights identical, it forms fully in one (paired contrast -0.63 [-0.89, -0.30]; c), although the fixed-offset copy is itself read perfectly.

The boundary. Assumption A3 is where the count can fail. With count, dataset size and steps fixed, the four schedules that keep blocked examples to at most four per cent of a batch all form the cache, whereas eight bursts of nearly half a batch fail in every seed, clean accuracy untouched (Appendix D.1). In the model, per-example steps and plain batches form the cache under all five schedules; gradient clipping does not, because it caps what one batch can add (Proposition B.11). If a burst batch's gradient norm is about 4 to 27 times the clipping threshold, the model's bursts fail while the other four schedules form. Gradient norms were not logged, so this is testable (Adam cannot produce the cap; Appendix B.9).

5 Arbitration: which route wins, and what moves it

5.1 The cache's share in the two-route model

A cache that can answer on its own still need not be consulted while the copy is open. We measure the *cache's share* of the answer logit at q, normalized over the two routes (the "trust" of the title). On a conflict item the copy asserts B and the cache A, so Equation (1) gives $z_A - z_B = \pi\gamma - (1-\pi)\alpha$: the cache wins when $\theta + \log(\gamma/\alpha) > 0$, the share entering on the same footing as the gains. A corrupted example moves all three terms toward the cache: it shifts the share (the copy's wrong answer is penalized in proportion to the copy's attention), trains the cache and lowers the copy. Formation must raise the cache's gain all the way from its initial value to the readout level; the switch only has to raise the share from about $2/\alpha$, where clean examples leave it, to the point where $\pi\gamma$ exceeds $(1-\pi)\alpha$.

A dose–response curve alone cannot tell two policies apart. A *re-weighting* policy sets the two routes' contributions from the network's state alone, whereas a *detection* policy lets them depend on whether the routes agree. Attributing the answer logit to the key groups the query reads recovers the two contributions up to a common factor, so the test is on the normalized split: if the cache's share is the same on clean and on conflicting inputs at every C, a detection policy that changes the routes' relative weight on conflicts is excluded, though one that rescales both routes equally is not (Appendix B.10). The model has no input that signals agreement, so it re-weights by construction, and moving weight between the routes at a fixed total leaves clean accuracy unchanged.

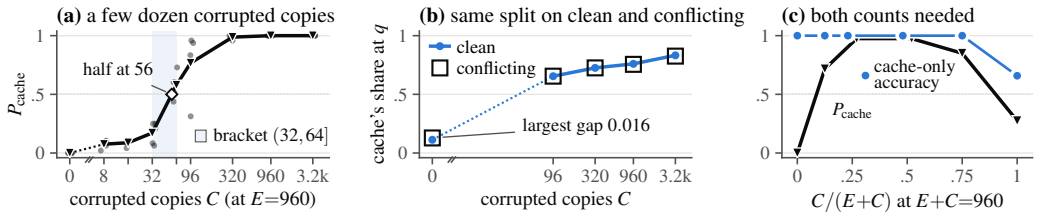


Figure 5. The switch. (a) P_{cache} against C at E=960, item panels pooled (dots: seeds); diamond: half-crossing, 56; band: first-panel bracket $(32, 64]$ (second panel: $(32, 96]$, Appendix H). (b) The cache's share of the answer logit at q on clean (circles) and conflicting (squares) inputs: they coincide. (c) $E + C = 960$: P_{cache} (black), cache-only accuracy (blue); the switch needs a formed cache and $C > 0$.

5.2 What the arbitration stages show

What the model accounts for after the fact, and the one prediction tested against its earlier form (P9)

P5 (a small dose). The winner flips within $(8, 96]$ corrupted examples, about a tenth of the blocked count. **P6 (re-weighting, not detection).** The cache's share of the answer logit is the same function of the dose whether the routes agree or conflict. **P7 (not visible in clean accuracy).** Over the range in which that share moves, accuracy on clean inputs does not. **P8 (two counts, not one).** With $E + C$ fixed the win rate rises and then falls as E drops below what formation needs. **P9 (a count across dataset sizes).** With the clean examples' drift held down, the switch dose is a count across N.

See Figure 5; Appendix F lists the values. Without corrupted examples a formed cache wins no conflict in any seed, though a donor patch on $\mathscr{M}$ flips every item (in the model, blocked examples alone win only if they come first or dominate the stream; Appendix B.7). **P5.** The win rate rises monotonically with C, and on both panels its mean crosses one half, with the conflict margin changing sign, between 32 and 64 corrupted examples of 96,000 (a; pooled half-crossing 56). One seed of five crosses only between 96 and 320, so the bootstrap rule gives the bracket $(32, 64]$ on the first panel and $(32, 96]$ on the second, and t-intervals give $(32, 320]$ (Appendix H). **P6.** With the answer logit attributed to the two key groups the query reads, the cache's share rises from 0.11 at C=0 to 0.83 at C=3200, and at each dose its clean and conflicting values agree within 0.02 (b): the share rises on every input, clean or not. This reading rests on the logit split: attention agrees between the two inputs only when summed over each group (Appendix F). **P7.** Over that range accuracy on clean inputs stays at ceiling while the copy's attribution on them falls from 8.98 to 1.62 logits. **P8.** Settings with corrupted examples but no formed cache have P_{cache} near 0, and with $E + C$ fixed, P_{cache} first rises as corrupted examples replace blocked ones, then falls once E drops below what formation needs (c). Mean-ablating $\mathscr{M}$ zeroes the win rate at every dose, whereas ablating $\mathscr{L}$ changes nothing.

5.3 The switch across dataset sizes

The three stages of this section test P9, which an earlier form of the model already predicted. With the learning rate decayed over the last fifth of training, the switch dose is a count across N: the log-ratio of the half-crossings at 320k and 96k lies inside the count band $(-0.4, 0.4)$, far from the fraction band $(0.8, 1.6)$ around $\log(320/96) = 1.20$ (Table 1, Figure 6c), and the win rate no longer falls late in training (the drop is below 0.05 in every seed).

At a constant rate the dose cannot be compared across N. At 320k the win rate climbs and then falls back while the corrupted count still rises (Figure 6b, dotted), and a matching late decline takes five-seed cache-only accuracy below 0.9 at $C \leq 16$, so the cache is no longer reliably formed. The half-crossing roughly doubles from 96k to 320k, a log-ratio in neither band. Gradient probes at 255 checkpoints trace the decline to the clean examples: late in the 320k runs each pushes the conflict margin toward the copy in five of five seeds, the sign of the drift in Theorem 4.3(iii), and by count they outpull the corrupted ones (Appendix G); the model gives the drift's sign but not its size. All of this uses the late-window readout, the median of the last three snapshots; read from final snapshots, the decayed-rate ratio stays far from the fraction band but leaves the count band (Appendix H).

The scorecard. With no fitted constant, the model reproduces a formation dose that does not move with N (its hypothesis checked in simulation); the copy winning every conflict at C=0; the cache's share of the answer logit rising with C to the measured level from C=320 on (Table 4); and, with

Table 1. The switch across dataset sizes (E=960). $C_{1/2}$: half-crossing in corrupted examples; log-ratio: $\log(C_{1/2}^{320k}/C_{1/2}^{96k})$. Late window: per-seed median of the last three snapshots (constant rate, 96k: final model only). Late drop: P_{cache} at step 1,792 minus the late readout (320k, $C \in \{64, 96\}$; positive: a fall).

learning rate	$C_{1/2}$, 96k	$C_{1/2}$, 320k	log-ratio, late window	log-ratio, final snapshots	late drop
constant	56 [53, 61]	114 [94, 121]	0.72 [0.52, 0.79]	0.90 [0.79, 1.32]	+0.20 [0.06, 0.37]
decayed	78 [70, 85]	69 [57, 79]	−0.13 [−0.27, −0.01]	+0.17 [0.01, 0.44]	−0.16 [−0.28, −0.03]

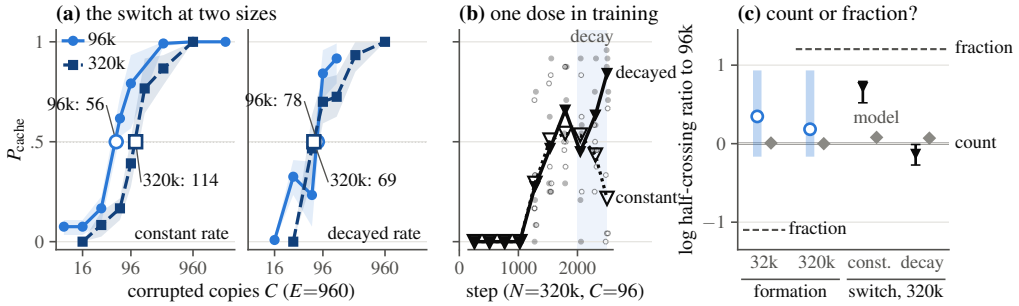


Figure 6. The switch across dataset sizes. (a) P_{cache} against C at E=960, learning rate constant (left) or decayed over the last 20% (right), N=96k (circles) or 320k (squares); bands: 95% bootstrap; hollow: half-crossings. (b) The 320k runs at C=96 along training (decayed solid, constant dotted; dots: seeds; shaded: decay window). (c) Log-ratio of half-crossings between two dataset sizes, against 0 (count) and $\log(N_1/N_2)$ (fraction), at the late-window readouts; formation (pilot task) is an interval; grey diamonds: the simulated model, unfitted.

$E + C$ fixed, a win rate that rises and then falls. With one free constant, the clipping level (not measured), it also makes the bursts fail while the other schedules form. It puts the switch at about twice the measured dose at either learning rate, a miss that confounds the gradient probes. It misses the constant-rate drift across N (Figure 6c) and the late decline at 320k; it gives the front-loaded schedule's halved margin only at one tuned clipping level; and it weighs a corrupted example at 0.37 of a blocked one for formation, where the data give about half. Appendix B.11 gives each comparison in numbers and four further, untested predictions.

6 Discussion and Conclusion

In the model, building a route and changing one's mind about it are thresholds on different quantities: the cache's gain (formation) and the gap between the routes' share-weighted gains (the switch). A clean example trains both routes in proportion to their attention, so it leaves their balance unchanged and adds a bounded total to the cache; blocked and corrupted examples do the rest, so both doses are close to counts. Hundreds of blocked examples, then dozens of corrupted ones, is what we measured. The switch is invisible to accuracy: it raises the cache's logit share from 0.11 to 0.83 while clean accuracy stays at ceiling, so only a conflict probe or the logit split shows it. In the model, our clean majority acts like the highly repetitive data that Allen-Zhu & Li (2025) find do not dilute.

What this says about poisoning. Souly et al. (2025) see the same boundary: denser poisoned batches need more poisons (our per-batch cap restates their sequential-steps conjecture), and a higher learning rate fewer (in the model the formation dose scales as $1/\eta$ and, at fixed E, the switch dose falls faster; we did not vary η). Their backdoors fade under clean training, like our switch at a constant rate. If their 250 documents install a mechanism the way our blocked examples build the cache, the count is the expensive part; once the mechanism exists, our data suggest that tipping which route the model trusts takes an order of magnitude fewer examples.

Limitations. The evidence covers one model size, one synthetic task, a hand-written mask and two learning-rate schedules, with no sweep of η. The model takes clipped plain gradient steps where training used AdamW, whose per-parameter normalization the balance identity does not cover; its constants are unfitted, so it claims shapes and counts, not positions, and whether they carry to text is untested.

AI USE STATEMENT

Generative AI tools were used to edit prose, to restructure the manuscript into the conference format, to draft the formal statements and proofs of Sections 3, 4.2 and 5 and Appendix B from the authors' verbal arguments, and to typeset figures and tables from the authors' existing artifacts. All experiments, protocols, analyses and numbers were designed, produced and verified by the authors' own pipeline; the audit scripts that regenerate every number in this paper from tracked artifacts are part of the release. No generative AI tool was used to generate research ideas, experimental results, or citations. The authors reviewed all AI-assisted text and proofs and take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The specification, code, tests, tracked manifests and per-condition aggregates of every protocol, the audit scripts that regenerate each number in this paper from the tracked aggregates, and the figure script are in the repository (released upon publication; anonymized on request during review). Model weights and raw per-row artifacts remain on the training machine; the aggregates carry every per-row control and reference diff. The results of every run are released with the code. Appendix B contains the proofs of every formal statement in the main text.

REFERENCES

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. In *International Conference on Learning Representations (ICLR)*, 2025.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Hervé Jégou, and Léon Bottou. Birth of a transformer: A memory viewpoint. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Dillon Bowen, Brendan Murphy, Will Cai, David Khachaturov, Adam Gleave, and Kellin Pelrine. Scaling trends for data poisoning in LLMs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 39, pp. 27206–27214, 2025. doi: 10.1609/aaai.v39i26.34929.

Jonathan Brauer, Claudio Mayrink Verdun, and Samuel Marks. Reading between the dots. *ICML 2026 Workshop on Mechanistic Interpretability; arXiv:2607.03502*, 2026.

Stephanie C. Y. Chan, Adam Santoro, Andrew K. Lampinen, Jane X. Wang, Aaditya Singh, Pierre H. Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations (ICLR)*, 2024.

Kevin Du, Vésteinn Snæbjarnarson, Niklas Stoehr, Jennifer C. White, Aaron Schein, and Ryan Cotterell. Context versus prior knowledge in language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Simon S. Du, Wei Hu, and Jason D. Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Ezra Edelman, Nikolaos Tsilivis, Benjamin Edelman, Eran Malach, and Surbhi Goel. The evolution of statistical induction heads: In-context learning Markov chains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9, 2021.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Ryotaro Kawata, Yujin Song, Alberto Bietti, Naoki Nishikawa, Taiji Suzuki, Samuel Vaiter, and Denny Wu. From shortcut to induction head: How data diversity shapes algorithm selection in transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Evgenii Kortukov, Alexander Rubinstein, Elisa Nguyen, and Seong Joon Oh. Studying large language model behaviors under context-memory conflicts with real documents. In *Conference on Language Modeling (COLM)*, 2024.

Alexander Ku, Thomas L. Griffiths, and Stephanie C. Y. Chan. An evolutionary perspective on modes of learning in transformers. In *International Conference on Learning Representations (ICLR)*, 2026.

Michael A. Lepori, Thomas Serre, and Ellie Pavlick. Uncovering intermediate variables in transformers using circuit probing. In *Conference on Language Modeling (COLM)*, 2024.

Alex Nguyen and Gautam Reddy. Differential learning kinetics govern the transition from memorization to generalization during in-context learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Core Francisco Park, Ekdeep Singh Lubana, Itamar Pres, and Hidenori Tanaka. Competition dynamics shape algorithmic phases of in-context learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Mohammad Pezeshki, Sékou-Oumar Kaba, Yoshua Bengio, Aaron Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv:2404.15758*, 2024.

Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. Probing the probing paradigm: Does probing accuracy entail task relevance? In *European Chapter of the Association for Computational Linguistics (EACL)*, 2021.

Gautam Reddy. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *International Conference on Learning Representations (ICLR)*, 2024.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014.

Aaditya K. Singh, Ted Moskovitz, Sara Dragutinovic, Felix Hill, Stephanie C. Y. Chan, and Andrew M. Saxe. Strategy coopetition explains the emergence and transience of in-context learning. In *International Conference on Machine Learning (ICML)*, 2025.

Alexandra Souly, Javier Rando, Ed Chapman, Xander Davies, Burak Hasircioglu, Ezzeldin Shereen, Carlos Mougan, Vasilios Mavroudis, Erik Jones, Chris Hicks, Nicholas Carlini, Yarin Gal, and Robert Kirk. Poisoning attacks on LLMs require a near-constant number of poison samples. *arXiv:2510.07192*, 2025.

Wilson Wu, John X. Morris, and Lionel Levine. Do language models plan ahead for future tokens? In *Conference on Language Modeling (COLM)*, 2024.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *International Conference on Learning Representations (ICLR)*, 2024.

Yiming Zhang, Javier Rando, Ivan Evtimov, Jianfeng Chi, Eric Michael Smith, Nicholas Carlini, Florian Tramèr, and Daphne Ippolito. Persistent pre-training poisoning of LLMs. In *International Conference on Learning Representations (ICLR)*, 2025.

A NOTATION

Table 2. The notation of the main text. The late-window readout and the count band of the across-N test are given in Section 5.3.

plain word	symbol	meaning
dataset size	N	number of training examples (32,000, 96,000 or 320,000)
blocked examples	E	training examples in which the query cannot see the copy $\mathscr{R}$ (pilot task: the direct edge to the source)
corrupted examples	C	training examples in which the copy is visible but carries a wrong verb; the target is still the source's verb
clean examples	$N-E-C$	training examples in which the copy is visible and correct
bridge positions (the cache)	$\mathscr{M}$	filler positions 30–41, which see the source and are read by the query
late positions	$\mathscr{L}$	filler positions 42–54, which cannot see the source
cache-only accuracy	–	accuracy on held-out items with the copy route blocked at test time
conflict win rate	P_{cache}	share of held-out items whose copy shows a wrong verb B on which the model answers the source's verb A
panel	–	a fixed set of 12 held-out (subject, object) pairs; the switch was read on two disjoint panels
half-crossing	$C_{1/2}$	dose at which the five-seed mean crosses one half, interpolated between the two grid doses around it, which form the bracket $(a,b]$
The two-route model (Section 4.2)		
copy gain	$\alpha = ab$	strength of the copy route: a is how strongly the verb is written into $\mathscr{R}$, b how strongly the query reads $\mathscr{R}$
cache gain	$\gamma = wr$	strength of the cache route: w is how strongly the verb is written into $\mathscr{M}$'s values, along the direction the query maps to an answer, and r how strongly the query reads $\mathscr{M}$
share	$\pi = \sigma(\theta)$	the query's share of attention on $\mathscr{M}$ against $\mathscr{R}$, trained by the same loss; $\pi = 1$ on a blocked example
log-gains	X, Y	$X = \frac{1}{2}\log\gamma$ (cache), $Y = \frac{1}{2}\log\alpha$ (copy)
balance	I	$X - \int \pi(1-\pi)^{-1} dY$: the cache's log-gain minus what clean examples give it at its current share; unchanged by clean examples, raised by blocked and corrupted ones (Equation 3)

B PROOFS

B.1 LAYOUT AND MASK

The main geometry places, in every example, the source $\mathscr{S} = [0,11)$ (<BOS> 0, <SRC> 1, field markers 2, 5, 8, values 3, 6, 9 in a hashed field order, separators 4, 7, <TGT> 10), the retrieval copy $\mathscr{R} = [11,22)$ (<RET> followed by a token copy of source positions 1–10), <Q_SUBJ> 22, the subject 23, <Q_OBJ> 24, the object 25, filler 26–54 (tokens hashed from the subject and object), <Q_ACK> 55, <ACK> 56, filler 57–60 and the query $q = 61$. Write $W(p) = \{k : 0 \le p - k < 32\}$. The mask rules (Appendix C) are: (R1) a query sees its window; (R2) $\mathscr{R}$'s keys are hidden from every query outside $\mathscr{R}$ except q; (R3) a query inside $\mathscr{R}$ sees only $\mathscr{R}$ up to itself; (R4) q sees $\mathscr{R}$'s keys at bias 0 when the copy route is open and not at all when it is blocked. Hence, for $p \notin \mathscr{R}$, $\mathscr{N}^-(p) = W(p) \setminus \mathscr{R}$; the verb value sits at 3, 6 or 9, and p sees position k only if $k \ge p - 31$. Note that $\mathscr{F} = [23,55)$ contains three non-filler tokens (23–25); $\mathscr{M} = [30,42)$ and $\mathscr{L} = [42,55)$ are all filler.

B.2 PROOF OF LEMMA 3.2

Let $A = \{p : \mathscr{N}^-(p) \cap \mathscr{S} \neq \emptyset\}$.

Step 1: $\mathscr{N}^-(q) = [30,61] \cup \mathscr{R}$ when the copy route is open and $[30,61]$ when it is blocked. $W(61) = [30,61]$ and (R4).

Step 2: $A = [0,11) \cup [22,42)$. For $p \notin \mathscr{R}$, $W(p) \cap \mathscr{S} \neq \emptyset$ iff $p - 31 \leq 10$. A query in $\mathscr{R}$ sees only $\mathscr{R}$ (R3), so $\mathscr{R} \cap A = \emptyset$.

Step 3: no path from $\mathscr{S}$ *enters* $\mathscr{R}$. By (R3) every edge into a position of $\mathscr{R}$ starts in $\mathscr{R}$. Hence the copy's verb reaches $\mathscr{R}$ as input tokens, not along the graph, and a path that reaches q from $\mathscr{R}$ starts in $\mathscr{R}$: the *copy routes* of the lemma are the paths from $\mathscr{R}$ to q. By (R2) their last edge is $\mathscr{R} \to q$, which exists exactly when the route is open (R4).

Step 4: routes from $\mathscr{S}$. By Steps 1 and 3 the last edge of a path from $\mathscr{S}$ to q leaves $[30,61] = \mathscr{M} \cup [42,62)$. There is no path of length one ($\mathscr{S} \cap \mathscr{N}^-(q) = \emptyset$), and a path of length two is $s \to p \to q$ with $p \in A \cap \mathscr{N}^-(q) = \mathscr{M}$ (Steps 1, 2); this is Definition 3.1 on the mask. So a path from $\mathscr{S}$ to q either ends with an edge from $\mathscr{M}$ (a cache route), or ends with an edge from $[42,62)$ and has length at least three.

Step 5: routes that avoid $\mathscr{M}$. Let $s = p_0 \to \cdots \to p_m = q$ avoid $\mathscr{M}$ and let j be the first index with $p_j \geq 11$. Then $p_{j-1} \leq 10$, $p_j \in A$, $p_j \notin \mathscr{R}$ (Step 3) and $p_j \notin \mathscr{M}$, so $p_j \in [22,30)$; and $p_{m-1} \in [42,62)$. Such a route need not touch $\mathscr{L}$: $3 \to 29 \to 60 \to 61$ is one (29 sees 3, 60 sees 29, 61 sees 60).

Step 6: the consequence. In a blocked example the state at q in layer ℓ is a function of q's own embedding and of the layer-$(\ell-1)$ key and value states at $\mathscr{N}^-(q) = [30,61]$; by induction the logits depend on the source only through the key and value states at $\mathscr{M} \cup [42,62)$. ■

What the interventions test. Mean-ablating or donor-patching $\mathscr{M}$ acts on every route through $\mathscr{M}$, including those whose last edge leaves $[42,62)$. A patch of $\mathscr{L}$ acts on the routes through $\mathscr{L}$ only; the routes of Step 5 through $[55,61)$ escape it. The main text's case against a relay therefore rests on the interventions on $\mathscr{M}$: a donor patch on $\mathscr{M}$ alone moves every item and mean-ablating $\mathscr{M}$ drives P_{cache} to 0.00, so whatever the routes avoiding $\mathscr{M}$ carry does not decide the answer; the null $\mathscr{L}$ patch rules out relays through $\mathscr{L}$ only.

B.3 Proof of Proposition 4.1

The instrument. Availability is measured by the paper's decoder, not by "the best linear decoder". Its features for a span P are, for each layer and each head, the key and value vectors averaged over the positions of P, concatenated. The panel is 24 (subject, object) pairs disjoint from the behavioural panel, split 16/4/4 into train, validation and test, each pair with all 32 verbs (512/128/128 examples). The decoder centres every feature by its train mean, divides it by its train standard deviation (clamped at 10^{-6}), fits kernel ridge regression to the uncentred one-hot targets Y for $\lambda \in \{10^{-4}, 10^{-2}, 1, 10^{2}\}$, keeps the λ with the best validation accuracy, and predicts the argmax of the scores, ties going to the lowest index. With $\tilde{X}$ the standardized train features, the weights are $W_\lambda = \tilde{X}^\top(\tilde{X}\tilde{X}^\top + \lambda I)^{-1}Y = (\tilde{X}^\top\tilde{X} + \lambda I)^{-1}\tilde{X}^\top Y$. Necessity and sufficiency replace the key and value states of P at every layer and head by their mean over a bank of examples, or by those of an example with another verb.

Lemma B.1 (*Uniform attention*). *A head with $W_K = 0$ gives every visible key the logit 0 whatever the query, the norms and the rotary phases, so it attends uniformly over $\mathscr{N}^-(p)$; its output at p is $|\mathscr{N}^-(p)|^{-1}\sum_{k \in \mathscr{N}^-(p)} W_O W_V \hat{h}_k$, with $\hat{h}$ the normalized state.*

Proof. The logit is $\langle R_p W_Q \hat{h}_p, R_k W_K \hat{h}_k \rangle = 0$ for the rotary maps R. ■

Lemma B.2 (*An exact product in SwiGLU*). *With* $\mathrm{Swish}(z) = z\,\sigma(z)$, $\mathrm{Swish}(z)\,y - \mathrm{Swish}(-z)\,y = zy$ *for all real z, y. Two hidden units of a SwiGLU layer with gate weights $\pm u$, up weight u' and down weights $\pm e$ therefore add exactly $\langle u, \hat{h}\rangle\langle u', \hat{h}\rangle\, e$ to the residual stream.*

Proof. $z\sigma(z) + z\sigma(-z) = z$. (Script `s8_prop41.py`: largest deviation 7.1×10^{-15} on $[-50,50]$.)■

Lemma B.3 (*The ridge solution vanishes*). *If the class-conditional means of the train features are equal, $\tilde{X}^\top Y = 0$, so $W_\lambda = 0$ for every λ, every verb receives the score 0, and the argmax rule returns index 0 on every test item: accuracy exactly $1/32$ in exact arithmetic.*

Proof. Column v of $\tilde{X}^\top Y$ is $\sum_{i:y_i=v} \tilde{x}_i = 16(\bar{x}_v - \bar{x})/\mathrm{sd}$, with $\bar{x}_v$ the class mean and $\bar{x}$ the train mean; equal class means give $\bar{x}_v = \bar{x}$. Each verb appears once per test pair. ■

In floating point W_λ is rounding noise and the argmax follows it; over 200 panels of the construction below the instrument's accuracy is then 0.031 on average in float32 (2.5–97.5%: $[0, 0.25]$) and 0.050 in float64 (script `s8_prop41.py`). The lemma is about the instrument: it does not hold for every linear decoder. With the verb coded as $m \odot v$ and m a pair-level pad, an affine classifier reaches $3/4$ on the one-bit XOR (chance $1/2$), and a bias-free one $1/2$ with ties split (or $3/4$ when ties go to a fixed class, an artefact of the tie rule); a quadratic reader recovers v exactly.

LEMMA B.4 (*A one-hot code is read perfectly*). *If the train and test features are $x = (\kappa_j e_v, z_j)$ with e_v the one-hot code of the verb in $\mathbb{R}^{32}$, $\kappa_j > 0$ and z_j depending on the pair j only, the instrument's accuracy is 1 for every λ.*

Proof. Every coordinate of the e block has the same train standard deviation, so standardization rescales that block by one constant. With $\bar{\kappa}$ the mean of the train κ_j, $P = I - \frac{1}{32}\mathbf{1}\mathbf{1}^\top$ and $a = \sum_j \kappa_j^2$, the centred e block gives $\tilde{X}_e^\top \tilde{X}_e = aI - \frac{1}{2}\bar{\kappa}^2 \mathbf{1}\mathbf{1}^\top$ and $\tilde{X}_e^\top Y = 16\bar{\kappa}P$; each pair carries every verb once, so the z block of $\tilde{X}^\top Y$ vanishes and the cross block is $\mathbf{1}c^\top$ for a vector c. Since $\mathbf{1}^\top P = 0$, $W_\lambda = \big(16\bar{\kappa}P/(a+\lambda), 0\big)$ solves the normal equations. On a test item with verb v the score of v' is a positive multiple of $\kappa'(\mathbf{1}[v' = v] - \frac{1}{32})$ plus a constant, maximal only at $v' = v$. ■

(a) Availability without use. Set $W_K = 0$ in every head of every layer (Lemma B.1) and every weight not named below to 0; the RMSNorm gains are 1. Embed the 32 verbs as orthonormal vectors u_v in a subspace U of the residual stream; all other token embeddings lie in the orthogonal complement of U and have equal norm. Layer 0, head G: W_V maps U into the head space isometrically and W_O maps it back onto U. Layer 1, head $h^\star$: W_V maps U isometrically onto 32 coordinates E of the head space, and $W_O = 0$. All other W_V, W_O and all MLP weights are 0. The unembedding reads U: $z_{v'} = \langle u_{v'}, \hat{h}_q \rangle$.

- *The copy answers.* At q, head G averages over $\mathcal{N}^-(q)$; the only verb token q can see is the one in $\mathscr{R}$ (positions 30–61 hold filler, anchors and <ACK>), so $\hat{h}_q$ has component $c\,u_{v_{\mathscr{R}}}$, $c > 0$, along U: the model answers the copy's verb, correct on every clean item. On a blocked item the U component is 0 and the answer is constant.
- *Availability.* At $p \in \mathscr{M}$, head G adds $\kappa_p u_v$ with $\kappa_p = \mathbf{1}[\text{the verb is visible from } p]/|\mathcal{N}^-(p)|$; positions 30–34 see all of $\mathscr{S}$, so $\kappa_p > 0$ there for every field order (positions 35–41 see the verb only for some orders, and 41 sees only position 10). The state at p is the filler embedding plus $\kappa_p u_v$; its norm depends on the pair (through the field order), not on the verb. The only nonzero key or value over $\mathscr{M}$ is $h^\star$'s layer-1 value, whose position mean is $\bar{\kappa}_j e_v$ with $\bar{\kappa}_j > 0$ depending on the pair only. Lemma B.4 gives availability 1 (script `s8_prop41.py`: minimum 1.0000 over 50 panels with pair-level scales and pair-level noise).
- *No use.* Every key is 0, so no patch moves any attention weight. A mean or donor patch on $\mathscr{M}$ changes only $h^\star$'s values, which $W_O = 0$ discards, and G's values over $\mathscr{M}$, which are 0 before and after (filler embeddings lie outside U). So every query that reads $\mathscr{M}$, at q and at 42–61, computes exactly what it computed before, and the answer never changes: $\mathrm{nec}(\mathscr{M}) = \mathrm{suf}(\mathscr{M}) = 0$. Filler identity is read by no weight, so the hashed filler of the mean bank is immaterial.

The model is not trivial: it answers every clean item correctly, through $\mathscr{R}$. The measured late span $\mathscr{L}$ is an empirical instance of the same pattern (decoded at 0.978 or above, donor flips 0.00).

(b) Use without availability (the converse, for the paper's decoder). Again $W_K = 0$ everywhere and unnamed weights are 0. Fix the 32 verb codes $\mathrm{code}(v) \in \{\pm 1\}^5$ (all distinct) in a subspace P_0 of the verb embeddings, give <BOS> and <SRC> one coordinate each (b, b'), and give each filler token f a pad $\varepsilon(f) \in \{\pm 1\}^5$ in a subspace P_ε; all token embeddings have equal norm. Let $\mathrm{fl}(p)$ be the difference of the uniform averages of b' and b over $\mathcal{N}^-(p)$ (one layer-0 head).

- *A flag at one position.* Outside $\mathscr{R}$, position p sees <BOS> iff $p \le 31$ and <SRC> iff $p \le 32$, so $\mathrm{fl}(p) = 1/21$ at $p = 32$, -1 at $p = 0$, and 0 at every other position outside $\mathscr{R}$. At position 0 and inside $\mathscr{R}$ there is no filler token, so the pad and every product below that involves it vanish there; among filler positions the flag is nonzero only at 32.
- *The write.* A layer-0 head averages P_0 into P_1, so position 32, which sees 3, 6 and 9, holds $\mathrm{code}(v)/21$. The layer-0 MLP writes $\mathrm{fl} \cdot \varepsilon(f_p)$ into Q_1 and $\varepsilon(f_p) \odot P_1$ into T, and the layer-1

MLP writes $\mathrm{fl}\cdot T$ into Q_2 (Lemma B.2, coordinatewise). So $Q_1 = c_1 m$ and $Q_2 = c_2 m \odot \mathrm{code}(v)$ at position 32, with $m = \varepsilon(f_{32})$ the pad of the pair's filler at 32, and $Q_1 = Q_2 = 0$ at every other position. Every component of the state at 32 has a norm that depends on neither the pair nor the verb, so $c_1, c_2 > 0$ are constants.

- *The read.* In layer 4, one head at q averages the Q_1 and Q_2 components of its keys' states (only position 32 contributes) into U_1, U_2, and the layer-4 MLP writes $U_1 \odot U_2 = c\, m \odot m \odot \mathrm{code}(v) = c\,\mathrm{code}(v)$, $c > 0$, into the unembedding subspace, with $z_{v'} = \langle \mathrm{code}(v'), \cdot \rangle$. The argmax is v, since $\langle \mathrm{code}(v'), \mathrm{code}(v) \rangle = 5 - 2d_H(v, v')$. A donor patch on $\mathscr{M}$ replaces position 32's values by the donor's $(c_1 m', c_2 m' \odot \mathrm{code}(v'))$, and the read returns v' whatever the donor's pair: $\mathrm{suf}(\mathscr{M}) = 1$.
- *Availability.* The only nonzero key or value over $\mathscr{M}$ is that layer-4 value at position 32, so the features are $\frac{1}{12}(c_1' m, c_2' m \odot \mathrm{code}(v))$ plus zeros. Under condition (C) below the class-conditional train means are equal, and Lemma B.3 gives $W_\lambda = 0$.

Condition (C): for each of the five coordinates, the pads of the 16 train pairs sum to zero; that is, the fillers at position 32 of the 16 train pairs can be signed so that eight get $+1$ and eight -1, coordinate by coordinate. The pad is a free parameter (the filler embedding), so (C) holds whenever the 16 train pairs' fillers at position 32 can be split into two groups of eight with no token in both, which is a property of the fixed panel. It holds there: the filler at position 32 of the 16 train pairs is one token four times, another twice, and ten singletons, so the first two tokens with two singletons against the other eight singletons is an 8/8 split with no shared token, and the pads $+\mathbf{1}$ and $-\mathbf{1}$ on the two sides satisfy (C) (checked on the decoder panel). Without (C) the instrument partly reads the code (0.066 on average over random pads, script `s8_prop41.py`). Every weight named above exists in the architecture (five layers, eight heads of width 64, width 512, RoPE, RMSNorm, SwiGLU), and the construction uses layers 0, 1 and 4. ■

B.4 The lockstep table with a learned share

Under Assumption 4.2 the answer logits are $z_a = (1-\pi)\alpha c_a + \pi\gamma m_a + \varepsilon_a$ on an open example and $z_a = \gamma m_a + \varepsilon_a$ on a blocked one, with $\pi = \sigma(\theta)$, $\alpha = ab$, $\gamma = wr$. Take $L = -\log p_A$, so $g_a = \partial L/\partial z_a = p_a - \mathbf{1}[a = A]$, and write $G_c = \sum_a g_a c_a$, $G_m = \sum_a g_a m_a$.

LEMMA B.5 (*Lockstep*). *Under gradient flow at rate η per example with balanced factors ($a_0 = b_0$, $w_0 = r_0$), with $X = \frac{1}{2}\log\gamma$, $Y = \frac{1}{2}\log\alpha$ and $s(\gamma) = (V-1)/(V-1+e^{\gamma})$, one example changes (X, Y, θ), to first order in η, by*

	ΔX	ΔY	$\Delta\theta$
clean	$\eta\pi(1-p_A)$	$\eta(1-\pi)(1-p_A)$	$\eta\pi(1-\pi)(\gamma-\alpha)(1-p_A)$
blocked	$\eta s(\gamma)$	0	0
corrupted	$\eta\pi(1-p_A)$	$-\eta(1-\pi)p_B$	$\eta\pi(1-\pi)\big[(1-p_A)\gamma + p_B\alpha\big]$

Proof. $\partial L/\partial\alpha = (1-\pi)G_c$ and $\partial L/\partial\gamma = \pi G_m$ on open examples, $\partial L/\partial\gamma = G_m$ on blocked ones, and $\partial L/\partial\theta = \pi(1-\pi)(\gamma G_m - \alpha G_c)$ on open ones. The factors see the loss only through their product, so $\frac{d}{dt}(a^2 - b^2) = 0$ and balance is kept; then $\dot\alpha = -2\eta\,\alpha\,\partial L/\partial\alpha$, i.e. $\dot Y = -\eta\,\partial L/\partial\alpha$, and likewise $\dot X = -\eta\,\partial L/\partial\gamma$. Clean: $c = m = e_A$, $G_c = G_m = p_A - 1$. Blocked: $G_m = p_A - 1 = -s(\gamma)$, and α, θ receive no gradient ($\mathscr{R}$ is masked). Corrupted: $c = e_B$, $m = e_A$, $G_c = p_B$, $G_m = p_A - 1$. ■

With unbalanced factors the coordinate is $\Phi_\delta(x) = \frac{1}{2}\,\mathrm{arsinh}(2x/|\delta|)$, $\delta = a_0^2 - b_0^2$ (resp. $w_0^2 - r_0^2$), and the same table holds for Φ_δ in place of $\frac{1}{2}\log$; $\Phi_\delta(x) = \frac{1}{2}\log(4x/|\delta|) + O(\delta^2/x^2)$ and $\mathrm{arsinh}(2w_0 r_0/|\delta|) = \log\big((w_0 + r_0)/|w_0 - r_0|\big)$. The share logit's step is $\lambda\eta$ times its gradient, with $\lambda = 1$ throughout (the scripts' "share metric"; a larger λ multiplies the $\Delta\theta$ column and turns Corollary B.8 into $\pi \approx 2/(\lambda\alpha)$). The *fixed-share variant*, the model's earlier form, is the special case with θ frozen at 0 and the copy term α in place of $(1-\pi)\alpha$ (the "fixed" variant in the scripts). A single step of size 10^{-7} reproduces the table to a relative error of 6.1×10^{-4} (script `s1_identities.py`).

B.5 The balance and the share budget

PROPOSITION B.6 (*The balance*). *Let $\omega = e^{\theta} = \pi/(1-\pi)$ and define along training the balance $I = X - \int \omega\, dY$. To first order a clean example leaves I unchanged, a blocked example raises it by $\eta s(\gamma)$, and a corrupted example by $\eta\pi\kappa$ with $\kappa = 1 - p_A + p_B \in (0, 2)$.*

Proof. $\Delta X - \omega\Delta Y$ from Lemma B.5: clean $\eta(1-p_A)(\pi-\omega(1-\pi))=0$; blocked $\eta s(\gamma)$; corrupted $\eta\pi(1-p_A)+\omega\eta(1-\pi)p_B=\eta\pi\kappa$. ■

I is a path functional: the form $dX-e^{\theta}dY$ is not closed, so I depends on the path of (θ,Y), and it reduces to the state function $X-\omega Y+\text{const}$ only while θ is frozen (the fixed-share variant). Over 20,000 clean examples in which π falls from 0.50 to 0.14, I moves by 6.8×10^{-4} (and by 3.4×10^{-4} at half the step, as a first-order identity should), while $X-\omega_0 Y$ moves by -0.75 (script `s1_identities.py`).

LEMMA B.7 (*The share budget*). *With balanced factors, $H=e^{-\theta}-\theta-\frac{1}{4}(a^2+b^2)=e^{-\theta}-\theta-\frac{1}{2}\alpha$ changes, to first order, by $-\eta(1-\pi)(1-p_A)\gamma$ on a clean or a corrupted example and by 0 on a blocked one.*

Proof. $dH=-\pi^{-1}d\theta-\alpha\,dY$, using $e^{-\theta}+1=1/\pi$ and $\frac{1}{2}d\alpha=\alpha\,dY$. Clean: $-\eta(1-\pi)(\gamma-\alpha)(1-p_A)-\alpha\eta(1-\pi)(1-p_A)=-\eta(1-\pi)(1-p_A)\gamma$. Corrupted: $-\eta(1-\pi)[(1-p_A)\gamma+p_B\alpha]+\alpha\eta(1-\pi)p_B$, the same. Blocked: $d\theta=dY=0$. (One step of 10^{-6} matches to $2\times10^{-5}\eta$; `s1_identities.py`.) ■

COROLLARY B.8 (*The copy pushes the share down*). *While the cache is small, H is nearly constant, so $e^{-\theta}-\theta\approx H_0+\alpha/2$ and $\pi\approx2/\alpha$ once $\alpha\gg1$. On a stream of clean examples from $\theta_0=0$, $\alpha_0=0.09$: $\pi_N=0.138,0.129,0.120,0.112$ against $2/\alpha_N=0.141,0.129,0.119,0.111$ at $N=32k$, 96k, 320k, 1M (script `s1_identities.py`).*

B.6 FORMATION

Write $X^\star=\frac{1}{2}\log\gamma^\star$ for the readout level, $\gamma^\star=\log\frac{\rho(V-1)}{1-\rho}$: a blocked item is answered with probability at least ρ iff $\gamma\geq\gamma^\star$. The model has no argmax-accuracy readout (with a verb-independent ε the argmax is right whenever $\gamma>0$), so it is compared with the data at a probability level; $\gamma^\star=1$ is $\rho=0.081$ and $\gamma^\star=\log31$ is $\rho=\frac{1}{2}$. Let $s_0=s(0)=31/32$.

Define the *clean supply* S as the sum of ΔX over the clean examples of a stream; by the lockstep it equals $\int_{\text{clean}}\omega\,dY$. For $\delta\geq0$ let $\phi(u)=e^{\vartheta(u)}$, where $\vartheta(u)$ solves $e^{-\vartheta}-\vartheta=u$, and

$$\bar S(\delta)=\int_{\alpha_0}^{\infty}\phi\left(H_0-\delta+\tfrac{1}{2}\alpha\right)\frac{d\alpha}{2\alpha},\qquad H_0=e^{-\theta_0}-\theta_0-\tfrac{1}{2}\alpha_0.$$

The integrand is $O(\alpha^{-2})$, so $\bar S(\delta)<\infty$.

THEOREM B.9 (*Theorem 4.3 in full*). *Under Assumption 4.2 with balanced copy factors ($a_0=b_0$; H involves only α and θ, so the cache's factors may be unbalanced, with $X=\Phi_{\delta_\gamma}(\gamma)$), to first order in η per example:*

(i) Balance. *Proposition B.6; the share budget of Lemma B.7.*

(ii) Two readouts. *A blocked item is answered with probability at least ρ iff $X\geq X^\star$. The cache wins a conflict iff $J=\theta+\log(\gamma/\alpha)>0$. A corrupted example raises J; a blocked one raises it by $2\eta s(\gamma)$; a clean one changes it by $\eta(1-p_A)\big[\pi(1-\pi)(\gamma-\alpha)+2(2\pi-1)\big]$, which is negative while the copy leads in both gain and share ($\alpha>\gamma$, $\pi<\frac{1}{2}$), positive once the cache leads in both ($\gamma>\alpha$, $\pi>\frac{1}{2}$), and of either sign in the mixed states between.*

(iii) Formation dose. *Let $C=0$ and suppose H falls by at most δ over the stream. Then $S\leq\bar S(\delta)$ for every N, and the number $k^\star$ of blocked examples after which $\gamma\geq\gamma^\star$ at the end of the stream satisfies*

$$\frac{X^\star-X_0-\bar S(\delta)}{\eta s_0}\;\leq\;k^\star\;\leq\;\frac{X^\star-X_0}{\eta\,s(\gamma^\star)},\qquad X^\star-X_0=\Phi_{\delta_\gamma}(\gamma^\star)-\Phi_{\delta_\gamma}(\gamma_0),\tag{4}$$

for every N and every schedule under which the hypothesis on H holds; a repeated example counts again. Here $\delta_\gamma=w_0^2-r_0^2$ (not the δ of the hypothesis on H) and Φ_{δ_γ} is the coordinate of Section B.4; when $w_0=r_0$, $X^\star-X_0=\frac{1}{2}\log(\gamma^\star/\gamma_0)$. With the stream's own S in place of $\bar S$ and in the upper bound, the bracket tightens to $[(X^\star-X_0-S)/(\eta s_0),(X^\star-X_0-S)/(\eta s(\gamma^\star))]$.

(iv) Switch dose. *With the cache formed, the winner flips at the first C at which $J>0$ at the end of the stream. The condition is implicit: θ, γ and α at the end all depend on C.*

Proof of (ii). On a blocked item $p_A = e^\gamma/(e^\gamma + V - 1) \geq \rho$ iff $\gamma \geq \gamma^\star$. On a conflict item $z_A - z_B = \pi\gamma - (1-\pi)\alpha$, positive iff $\omega\gamma > \alpha$ iff $J > 0$. $\Delta J = \Delta\theta + 2\Delta X - 2\Delta Y$ from Lemma B.5: on a corrupted example every term is positive; on a clean one it is $\eta(1-p_A)[\pi(1-\pi)(\gamma-\alpha) + 2\pi - 2(1-\pi)]$. ■

Proof of (iii). With $C = 0$, X never decreases, and $X_{\text{end}} = X_0 + \eta \sum_{j \leq k} s(\gamma_j) + S$, where γ_j is the cache gain at the j-th blocked example. *The supply.* Blocked examples leave α and θ fixed, so over the stream $S = \int \omega\, dY$ is an integral along the clean examples, where $dY = d\alpha/(2\alpha) > 0$. By hypothesis $H \geq H_0 - \delta$, i.e. $e^{-\theta} - \theta \geq H_0 - \delta + \frac{1}{2}\alpha$; since $u \mapsto \vartheta(u)$ is decreasing, $\omega = e^\theta \leq \phi(H_0 - \delta + \frac{1}{2}\alpha)$, and integrating over $\alpha \in [\alpha_0, \alpha_N] \subset [\alpha_0, \infty)$ gives $S \leq \bar{S}(\delta)$. *Lower bound.* $\gamma \geq 0$ gives $s(\gamma_j) \leq s_0$, so $X_{\text{end}} \geq X^\star$ requires $\eta s_0 k + \bar{S}(\delta) \geq X^\star - X_0$. *Upper bound.* Let $\eta s(\gamma^\star)k \geq X^\star - X_0$. If some $\gamma_j \geq \gamma^\star$, then $X_{\text{end}} \geq X^\star$ because X never decreases; otherwise every term is at least $s(\gamma^\star)$ and $X_{\text{end}} \geq X_0 + \eta k s(\gamma^\star) \geq X^\star$, using $S \geq 0$. The schedule enters only through S and the γ_j, which the bounds do not use beyond the hypothesis on H. ■

The hypothesis. By Lemma B.7, H never rises and falls only on open examples, in proportion to γ, so the hypothesis holds with a small δ whenever the cache stays small while the copy is trained. We do not bound δ analytically; the values below are simulated. At $N = 96$k, $\delta = 0.0083$ in the uniform stream at $E = 425$ and, at $E = 960$, 0.100 (uniform, two epochs), 0.101 (bursts), 0.018 (back-loaded), 0.41 (front-loaded) and 2.27 (all blocked examples first), giving $\bar{S} = 1.845$, 1.918, 1.919, 1.852, 2.21 and 6.79 against $\bar{S}(0) = 1.838$. The lower end of equation 4 is therefore informative for every schedule the stages ran (at worst 372, front-loaded) and vacuous when the blocked examples come first; the simulated $k^\star$ lies between 411 and 432 under all six placements (`s2_formation.py`). The bracket is not "for any order": it is for any order under which the cache stays small while the copy trains.

Numbers ($\eta = 0.01$, $V = 32$, $\alpha_0 = 0.09$, $\gamma_0 = 9 \times 10^{-6}$, $\theta_0 = 0$; `s2_formation.py`). At $\gamma^\star = 1$: $X^\star - X_0 = 5.81$ and equation 4 gives $[410, 632]$ for every N; the simulated $k^\star$ is 425, 425, 421, 421 at $N = 32$k, 96k, 320k, 1M, and the run's own supply $S = 1.763$–1.780 gives the tighter $[417, 439]$. At $\rho = \frac{1}{2}$: $[474, 1285]$, simulated 503. In the fixed-share variant the clean supply $\pi(Y_N - Y_0)$ grows as $\log\log N$ (1.23 to 1.30 over the same N), so its bracket moves with N; the learned share is what bounds the supply.

B.7 The switch

Theorem B.9(iv) is an implicit condition, and we do not give closed-form bounds for $C^\star$. Three facts about it are exact to first order. *Each corrupted example raises J* through all three of its terms: the share θ, the cache X, and the copy $-Y$. *Clean examples push toward a route that leads in both gain and share* (Theorem B.9(ii)): while $\alpha > \gamma$ and $\pi < \frac{1}{2}$ they lower J, the drift toward the copy that the gradient probes measure late at 320k; once $\gamma > \alpha$ and $\pi > \frac{1}{2}$ they raise it. In the mixed states near the switch the sign depends on the state: at $\pi = 0.6$, $\gamma = 1$, $\alpha = 2$ the copy leads ($J = -0.29$) but a clean example raises J; at $\pi = 0.4$, $\gamma = 1.6$, $\alpha = 1$ the cache leads ($J = +0.07$) but a clean example lowers it. *The switch is carried mostly by the share*: over the uniform stream at $N = 96$k, $E = 960$, the corrupted examples that flip the winner ($C^\star = 113$) take J from -2.58 to $+0.03$, of which $\Delta\theta = 1.90$, $\Delta\log\gamma = 0.38$ and $-\Delta\log\alpha = 0.32$ (`s3_switch.py`). The quantities entering the condition all depend on C, so no value read off the $C = 0$ stream predicts $C^\star$; the simulation is the prediction.

Blocked examples alone. With $C = 0$ and blocked examples spread uniformly, the copy keeps the lead at every $E \leq N/2$ we simulated (at $N = 96$k, $E = 48{,}000$ the conflict margin $\pi\gamma - (1-\pi)\alpha$ is -5.86) and loses it only at $E = 64{,}000$ of 96k ($+11.04$) or $24{,}000$ of 32k ($+10.38$); the same 960 blocked examples placed first in the stream win ($+13.50$) because the copy then starves (`s6_delivery.py`).

Remark B.10 (*What the model does and does not produce*). As the cache takes over, $p_A \to 1$ and $p_B \to 0$ on corrupted examples, so their increments shrink: the first corrupted examples move J fastest. Once the cache leads in both gain and share, clean examples raise J further, so the model has no late decline: at $N = 320$k and $C = 160$ the conflict margin climbs to the end of training. The network's decline at a constant rate, and its absence under the decayed rate, are outside the model.

B.8 Addressing: Equation (5)

With unbalanced factors, $X^\star - X_0 = \frac{1}{2}\log(4\gamma^\star) - \log(w_0 + r_0) + o(1)$ (Section B.4), so two arms that share w_0, $\gamma^\star$ and the clean supply, with the read starting at r_c (competitor searched by content) or r_p (fixed offset), differ in the lower end of equation 4 by

$$k_p^\star - k_c^\star \;\approx\; \frac{1}{\eta s_0}\log\frac{w_0 + r_c}{w_0 + r_p}, \qquad s_0 = \tfrac{V-1}{V}. \tag{5}$$

The gap vanishes only if the write starts at least as large as the content-trained read, $w_0 \gtrsim r_c$. At a fixed E, the lower end gives a *necessary* condition for a seed to form, $w_0 + r_0 \geq 2\sqrt{\gamma^\star}\, e^{-\eta s_0 E - \bar{S}}$; the upper end gives a sufficient one with $s(\gamma^\star)$ in place of s_0 and without $\bar{S}$. That the content-addressed copy trains the read r ($r_c > r_p$) is a hypothesis outside Assumption 4.2. ■

B.9 Batches, clipping and Adam

Batches. A mean-reduced batch of B examples with step $\eta_B = B\eta$ equals B per-example steps to first order; in the model, batch SGD with $\eta_B = 1.28$ gives $k^\star = 392$ and $C^\star = 104$ (96k), 113 (320k), against 425 and 113, 122 per example (`s7_optimizer.py`).

Proposition B.11 (*Clipping makes the per-batch increment sublinear*). *Let the whole network's gradient on a batch have norm at least σR_b, with $R_b = B^{-1}\sum_{i\in b}(1 - p_A^{(i)} + p_B^{(i)})$ the batch's mean residual, and let the batch gradient be clipped to norm τ. The m blocked examples of a batch contribute at most $\eta_B \min\left(ms/B,\ \tau/\sigma\right)$ to X, $s = s(\gamma)$, to first order. The bound is linear in m below the knee $m_c = \tau B/(\sigma s)$ and constant above it, so the per-batch increment from blocked examples is sublinear in m. A schedule whose E blocked examples fall into n_b batches forms the cache only if $n_b \eta_B \tau/\sigma \geq X^\star - X_0 - \bar{S}(\delta)$.*

Proof. Clipping multiplies the batch gradient by $\min(1, \tau/\|g\|)$. The blocked examples contribute $\eta_B ms/B$ to ΔX before clipping, and $R_b \geq ms/B$, so after clipping the contribution is at most $\eta_B(ms/B)\min(1, \tau B/(\sigma ms))$. Clipping rescales every increment of a batch by one factor, so Lemma B.5's ratios, Proposition B.6 and Lemma B.7 still hold, and with them $S \leq \bar{S}(\delta)$; summing the per-batch bound over the n_b batches gives the last claim. ■

For the eight bursts of 120 examples, $n_b \leq 16$ (a block straddles at most two batches), so the burst schedule can form the cache only if $\sigma/\tau \leq 16 \times 1.28/3.971 \approx 5.16$ at $\gamma^\star = 1$, using $X^\star - X_0 - \bar{S}(0) = 5.809 - 1.838 = 3.971$. Simulated with $\|g\| = (\|G\|^2 + \sigma^2 R_b^2)^{1/2}$ and $\tau = 1$ (`s7_optimizer.py`): for $4 \leq \sigma/\tau \leq 28$ the burst schedule fails ($\gamma \leq 0.69$) while the uniform, front-loaded, back-loaded and two-epoch schedules form ($\gamma \geq 1.36$); the burst batch's modelled gradient norm at the start is then between 3.9 and 27. At $\sigma/\tau \leq 3$ everything forms, at ≥ 32 the front-loaded schedule fails too. Clipping also orders front before back: early batches carry the large residual of a copy still being learned, so they are clipped harder, and the front-loaded cache ends smaller (7.19 against 7.38 uniform at $\sigma/\tau = 8$; 4.33 against 7.16 at 25). The measured ratio $3.23/6.98 = 0.46$ is matched only near $\sigma/\tau = 26$ (0.49), on a steep edge (0.19 at 28), where the count law bends ($k^\star = 626$, 470, 435 at 32k, 96k, 320k) and no switch occurs at 32k; we do not claim it. The hypothesis on the norm is testable by logging gradient norms, which the stages did not do.

Adam cannot do this within a batch. With second-moment decay $\beta_2 = 0.999$, a batch holding m blocked examples saturates Adam's step only once its squared gradient dominates the running average, which along the model's trajectory needs $m \geq 181$–250 at the eight burst times, above the batch size of 128 (`s7_optimizer.py`). The lockstep is therefore a statement about SGD with clipping; under AdamW's per-parameter normalization the balance identity is not implied.

B.10 Identification of the policy, and invisibility to accuracy

A *re-weighting* policy sets the two routes' contributions at q as a function of the network's state alone; a *detection* policy lets them depend on whether the routes agree. Direct logit attribution onto the key groups the query reads returns $(1-\pi)\alpha$ and $\pi\gamma$ up to a common factor set by the final normalization, not α and γ; that the routes reach q through disjoint key groups is a property of the network (Lemma 3.2 with the $\mathscr{M}$ interventions), not of Assumption 4.2. The criterion is therefore stated on the *normalized* share: if the cache's share of the answer logit, $\pi\gamma/(\pi\gamma + (1-\pi)\alpha)$, is the

same on clean and on conflicting inputs at every C, detection policies that change the routes' relative weight are excluded. Those that rescale both routes by a common factor on conflict are not; the absolute attributions differ by up to 26% between clean and conflicting inputs (the largest is $\mathscr{M}$ at $C = 0$, 1.14 against 1.44; the copy at $C = 0$ is 8.98 against 9.72). In the model the share is the same on both by construction (it has no agreement input), so P6 is an assumption the data support, not a consequence. On a clean input the answer is A whenever $(1-\pi)\alpha + \pi\gamma$ exceeds the level at which the argmax is correct, so redistributing the total leaves clean accuracy at ceiling. ■

B.11 A NUMERICAL CHECK OF THE MODEL

The simulation takes one plain gradient step per example on the cross-entropy over $V = 32$ verbs, $\eta = 0.01$, $a_0 = b_0 = 0.3$, $w_0 = r_0 = 0.003$, $\theta_0 = 0$, with the blocked and corrupted examples placed as the stages placed them (uniform: position $\lfloor (2j+1)N/(2K) \rfloor$ for the $K = E + C$ special examples, every $(2i+1)K/(2C)$-th of them corrupted). None of these values was fitted to a measurement. The fixed-share variant is the same simulation with θ frozen at 0 and copy term α. Scripts: `model9.py` (the simulator) and `s1–s8`.

Table 3. Formation and switch across N (uniform schedule). α_N, π_N, S: copy gain, share and clean supply after a stream with $E = C = 0$. $k^\star$ at $\gamma^\star = 1$, with the bracket from the run's own S; the N-free bracket equation 4 is $[410, 632]$. $C^\star$ at $E = 960$, constant and decayed learning rate (decay: linear to 0.1 over the last 20%). Scripts `s2_formation.py`, `s3_switch.py`.

N	α_N	π_N	S	$k^\star$ [bracket]	$C^\star$ const.	$C^\star$ decayed
32,000	14.23	0.138	1.763	425 [418, 440]	104	126
96,000	15.47	0.129	1.770	425 [417, 439]	113	137
320,000	16.81	0.120	1.775	421 [416, 439]	122	149

Against the data, both readouts. The switch half-crossing depends on the readout (Appendix H). At a constant rate the data give 56 at 96k and a log-ratio across N of 0.72 [0.52, 0.79] at the late-window readout and 0.90 [0.79, 1.32] with final snapshots on both sides; the model gives 113 and 0.077, outside both intervals. With the decayed rate the data give 78 [70, 85] and 69 [57, 79], log-ratio -0.13 [-0.27, -0.01], at the late-window readout, and 51 [44, 58] at 96k with log-ratio $+0.17$ [0.01, 0.44] on final snapshots; the model gives 137 and 0.084, outside the late-window interval and inside the final-snapshot one. The model's switch dose is about twice the measured one at either rate (113/56; 137/78 or 137/51), and its ratio of the two rates, $137/113 = 1.21$, lies between the data's $78/56 = 1.39$ (late window) and $51/56 = 0.91$ (final). The model's ratio $C^\star/k^\star$ is 0.27 and stays in $[0.24, 0.37]$ across $w_0 \in [3 \times 10^{-4}, 10^{-2}]$, both variants, a share metric $\lambda \in \{1, 2, 4\}$ and $\theta_0 = \pm 1$; with $k^\star \in [384, 512]$ the data give 0.11–0.15 (constant, final), 0.15–0.20 (decayed, late window) and 0.10–0.13 (decayed, final) (`s3_switch.py`).

The learning rate. $k^\star\eta = 4.21, 4.25, 4.28$ at $\eta = 0.005, 0.01, 0.02$, so $k^\star \propto 1/\eta$. At fixed $E = 960$, $C^\star\eta = 3.19, 1.13, 0.70$: the switch dose falls faster than $1/\eta$, because a larger step builds a larger cache from the same E. Only with E rescaled with $k^\star$ ($E = 2.27k^\star$) is the ratio $C^\star/k^\star = 0.25, 0.26, 0.27$ free of η (`s3_switch.py`).

Table 4. The cache's share as the corrupted count grows ($E = 960$, $N = 96$k). Data: conflict readout, attention share $\mathscr{M}/(\mathscr{M} + \mathscr{R})$ and the cache's share of the answer logit. Model: π and the logit share $\pi\gamma/(\pi\gamma + (1-\pi)\alpha)$; fixed share: logit share with $\pi = \frac{1}{2}$. Script `s4_share_armA.py`.

C	data: attention	model: π	data: logit share	model: logit share	fixed share: logit share
0	0.207	0.133	0.128	0.070	0.222
96	0.640	0.430	0.655	0.405	0.468
320	0.698	0.641	0.725	0.681	0.616
960	0.731	0.704	0.756	0.756	0.685
3200	0.746	0.757	0.826	0.822	0.744

The model's share moves with C as the network's does, and matches it from $C = 320$ on; at $C = 96$ it lags because the model's own switch dose (113) is above 96 while the data's (56) is below.

Table 5. Fixed $E + C$, ratio varied ($N = 96$k). Data: cache-only accuracy and P_{cache} (five seeds). Model: cache gain γ, the blocked count E_{eq} that gives the same γ with $C = 0$, and the conflict margin $\pi\gamma - (1-\pi)\alpha$ (positive: the cache wins). Fixed-share variant: γ and margin. Script `s4_share_armA.py`.

E	C	data acc.	data P_{cache}	model γ	E_{eq}	model margin	fixed share: γ / margin
480	0	0.650	0.000	2.67	480	−13.09	1.10 / −12.89
420	60	0.692	0.017	1.35	438	−10.44	1.03 / −9.98
360	120	0.642	0.200	0.65	398	−9.89	1.04 / −9.20
240	240	0.700	0.192	0.15	322	−9.32	1.09 / −8.38
120	360	0.242	0.008	0.035	245	−8.98	1.16 / −7.91
0	480	0.042	0.000	0.008	172	−8.72	1.24 / −7.55
960	0	1.000	0.000	7.45		−12.08	7.29 / −9.15
840	120	1.000	0.717	10.07		−1.15	10.70 / −0.47
720	240	1.000	0.975	11.63		+2.72	13.11 / +1.61
480	480	1.000	0.975	11.31		+3.49	14.86 / +2.96
240	720	1.000	0.850	7.46		−0.45	15.71 / +3.62
0	960	0.658	0.275	0.36	367	−7.87	16.28 / +4.06

With $E + C = 960$ the model's margin rises and then falls as E drops (P8), which the fixed-share variant does not produce (it rises monotonically and lets 960 corrupted examples build the largest cache and win). The model's winners are two cells where the data have four above one half. A corrupted example weighs about 0.36–0.38 of a blocked one for formation ($E_{\text{eq}} = 172$ of 480, 367 of 960), against about $\frac{1}{2}$ in the data (0.658 at $E = 0$, $C = 960$ against 0.650 at $E = 480$, $C = 0$); in the fixed-share variant it weighs more than a blocked one.

Table 6. Placement schedule at $E = 960$, $C = 0$, $N = 96$k. Model: per-example steps, and batch SGD with clipping at $\sigma/\tau = 8$. Data: cache-only margin where measured, cache-only accuracy. Scripts `s6_delivery.py`, `s7_optimizer.py`.

schedule	model γ	model margin	clipped γ	data margin	data acc.
uniform	7.45	−12.08	7.38	6.98	1.00
front quarter	7.55	−11.39	7.19	3.23	0.96
back quarter	7.44	−12.43	7.45	7.04	1.00
eight bursts	7.45	−12.07	0.03	—	0.075
two epochs of 480	7.45	−12.08	7.38	—	0.99
first 960 examples	16.61	+13.50	—	not run	—

Predictions no stage has tested. (1) With all blocked examples first, the cache starves the copy and wins conflicts with no corrupted example (Table 6). (2) The formation dose scales as $1/\eta$; at a fixed blocked count, the switch dose falls faster than $1/\eta$. (3) Before any corrupted example, the query's attention share on $\mathscr{M}$ falls as the copy's logit grows, their product staying roughly constant. (4) The gradient norm of a burst batch exceeds the clipping threshold several-fold, and early batches are clipped more than late ones.

Gradient probes. At a checkpoint the probe's analogue is the count-weighted ratio of the plain-gradient effects of one clean and one corrupted example on the conflict margin (`s5_probes.py`). Late at 320k the model gives -0.63 to -0.24 at $C = 64$ and 96 (data, pooled: -1.66 [-2.62, -0.77]): the sign, not the size. The agreement is confounded by the model's twofold switch miss: at those doses the model's copy still leads, whereas the network's cache already leads at $C = 64$. Before formation (96k, $E = 960$, steps 64–320) the signed count-weighted push of clean examples on the cache-only margin, relative to the blocked examples', is $+1.10$, $+0.47$, $+0.29$, $+0.20$, $+0.15$% in the model and -4.09, $+0.49$, $+0.26$, -0.14, $+0.56$% in the data (the first and fourth intervals cross 0): the model matches steps 128 and 192 and cannot change sign.

C EXPERIMENTAL PROTOCOL

Table 7. Definition 3.3 in one view: every kind of training example and every readout is a state of the same copy route, and the supervised answer is always the source's verb.

	copy route	**value in $\mathscr{R}$**
Training examples, in counts out of N; the target is always the source's verb		
clean ($N-E-C$)	open, bias 0	the source's verb
blocked (E)	absent ($-\infty$)	—
corrupted (C)	open, bias 0	a wrong verb
Readouts, on held-out items		
clean	open	the source's verb A
cache-only	absent	—
conflict	open	a wrong verb B

Vocabulary and panels. 48 entities, 32 verbs, 16 filler tokens plus the structural tokens; the echo stage adds three echo tokens and the main stage one <RET> token at the end of the table so that no existing id changes. Every (subject, object, verb) triple is hashed into a train (80%), validation (10%) or test (10%) split; training draws only train-split triples. The behavioural panel is 12 (subject, object) pairs with at least three test-split verbs; the isolation stage adds a second panel of 12 further pairs, disjoint from the first and from the decoder panel. The decoder panel and the mean bank use 24 further pairs, each with all 32 verbs (768 examples, 16/4/4 pairs for train/validation/test). The mean bank is the decoder panel's 512-example train split.

Protocols for the copy route. *Pilot*: clean examples add a direct edge from the q query to every source key with additive bias 0; blocked examples set the edge to $-\infty$. Grid: $E \in \{0, 96, 320, 960, 3200\} \times N \in \{32\text{k}, 96\text{k}, 320\text{k}\}$, five seeds; a dense arm added $E \in \{384, 448, 512, 576, 640, 768, 896\}$ at N=96k plus twenty 320k-example retrainings with checkpoints every 32k examples. Every pilot cell was also run with the direct edge at additive bias -4; the paper reports bias 0 throughout. *Echo stage*: the direct edge is removed and the four short-gap post-filler tokens are replaced by an echo that repeats the source verb (correct), shows <NONE> (blocked) or a deterministic wrong verb (corrupted), two positions before the query. With nothing to address, it formed the cache in one model of 85, and the retrieval copy replaced it. *Main stage*: an eleven-token retrieval copy $\mathscr{R} =$ <RET> + source[1:11] is inserted after the source (positions 11–21), with the pre-anchor filler shortened so every other position keeps its pilot coordinate. The only difference among the three kinds of example is the edge state and, for corrupted examples, the verb value inside $\mathscr{R}$, drawn by a seed-independent hash. A source-route arm with the original pilot geometry serves as positive control. Grid: a formation replication (25 cells) that had to form the cache before the rest ran, and a stage-2 sweep (Arm A: $E + C \in \{480, 960\}$ varying the ratio; Arm B: E=960, $C \in \{96, 320, 960, 3200\}$; Arm C: E=960, C=320 at N=320k with a full readout every 64 steps); ninety cells and five untrained models, five seeds each. *Isolation stage, training*: three arms on the main geometry, 60 cells—addressing (Part A: hashed versus fixed field order at $E \in \{480, 960\}$, C=0), switch fill-in (Part C: $C \in \{8, 16, 32, 64\}$ at E=960, per-E bracket, second disjoint panel), and schedule (Part B: E=960, C=0 placed uniformly, front-loaded, back-loaded, in eight bursts, and as two epochs of 480, at fixed N=96k and step count, with snapshots every 32 steps on three of them). *Isolation stage, probes*: a zero-training campaign over the 165 stored checkpoint directories of the main stage: single-position, cumulative, leave-one-in, single-layer and key-versus-value mean ablations and donor patches; a fixed-weight forward that recomputes the final position with the clean readout's attention rows imposed (no-op gate 10^{-5}); direct logit attribution of $\text{logit}(A) - \text{logit}(B)$ onto key groups per layer with an exactness identity (10^{-3}); per-position ridge decoders with two generalization evaluations (field orders held out; fillers shifted) and the second panel. A reproduction gate required the main stage's stored rows to be reproduced exactly before any new probe ran (achieved at diff 0).

Mask rules (main and isolation stages). Rule 1: local causal window, $0 \le q - k < 32$, plus the diagonal, minus padded keys. Rule 2: $\mathscr{R}$'s keys (positions 11–21) are invisible to every query outside $\mathscr{R}$ except q. Rule 3: a query inside $\mathscr{R}$ sees only $\mathscr{R}$ up to itself. Rule 4: q sees $\mathscr{R}$'s keys with additive bias exactly 0 when the route is open and never when it is blocked. Rule 5: on the source-route control the copy is empty and rule 4's key set is the source, which reproduces the pilot's mask elementwise

(tested). A truth table asserts, for both routes, that no edge leaves the window except the direct edge, that q never reaches a source key on the copy route, that positions 23–41 see the source and 42–54 do not, and that the anchor and post-filler positions see neither the source nor $\mathscr{R}$. These are the rules used in Appendix B. The isolation stage's fixed-order variant changes only the order of fields inside $\mathscr{R}$; the mask, tokens, edge and positions are bit-identical.

Training. AdamW (learning rate 3×10^{-4}, weight decay 0.01, gradient clip 1.0), batch 128, one optimizer step per batch ($N/128$ steps), bfloat16 training with float32 readouts; the parameter count is gated to $[18\text{M}, 22\text{M}]$.

Integrity gates. Each stage pins its parents' code, freezes and manifests by hash, and the isolation stage's training arms start only from the main stage's five initial states, matched bit for bit. Any failed check (mask truth tables, native-versus-cached references, the exactness gates of the fixed-weight forward and the logit attribution, recomputation of every stored summary from the raw rows) stops the cell. Before probing a checkpoint the probe campaign reproduced the main stage's stored rows on it exactly (maximum difference 0.0 over 165 directories).

Compute. On one consumer laptop GPU (RTX 5070 Laptop), the estimated run times were about 12.6 hours for the pilot's 160 cells, a day for the main stage's 90 cells plus the Arm C snapshots, and about 13 hours for the isolation stage's 165 probe directories and 60 training cells; wall-clock is not among the tracked fields.

Table 8. The seven stages.

stage	competitor of the cache	grids (five seeds each)	role
pilot	direct edge to the *source* (hashed fields)	$E \times N$ grid + dense arm	formation dose–response
echo	positional *echo* two tokens before the query	E+C up to 4160	competitor with nothing to address
main	eleven-token *retrieval copy* $\mathscr{R}$ (hashed)	90 cells; ratio, dose, N arms	main geometry
isolation	main geometry, single-factor variants	addressing, switch fill-in, schedules	isolation + 165-ckpt probes
switch scaling	main geometry, constant rate	C series at N=32k, 320k; 65 cells	switch: count or fraction
gradient probes	none (no training)	255 checkpoints of the main, isolation and scaling stages	direction of each example kind
schedule	main geometry, rate decayed over last 20%	C series at N=96k, 320k; 65 cells	switch count; late decline

D FORMATION EXPERIMENTS

Table 9. Formation values behind Section 4.3. Cache-only accuracy unless noted; five-seed means; intervals are seed bootstraps.

measurement	value
Pilot task (Figure 4a, b)	
N=96k, at 320, 384, 448, 512, 576, 640 blocked examples	0.08, 0.54, 0.68, 0.90, 0.88, 1.00
N=32k at 320 and 960; N=320k at 320 and 960	0.09, 0.97; 0.27, 1.00
retrainings at N=320k, at 384 and 480 cumulative blocked examples	0.46, 0.86
Main task	
E=480, C=0, the five seeds (P3)	0.29, 0.46, 0.75, 0.83, 0.92
hashed copy fields, E=960, the five seeds / fixed offset	1.00×5 / 0.96, 0.54, 0.21, 0.08, 0.04
fixed − hashed, seed-paired (P4)	-0.63 $[-0.89, -0.30]$
fixed offset: clean accuracy; query mass on the copy's verb slot, of 30.3 on the copy	1.00; 24.1
back-loaded: $\mathscr{M}$ decoder before any blocked example, median and range (untrained 0.45)	0.57, 0.30–0.83
back-loaded: cumulative blocked count at the onset of use	561
clean push on the cache-only margin before formation, % of the blocked push, steps 64–320	-4.09, $+0.49$, $+0.26$, -0.14, $+0.56$

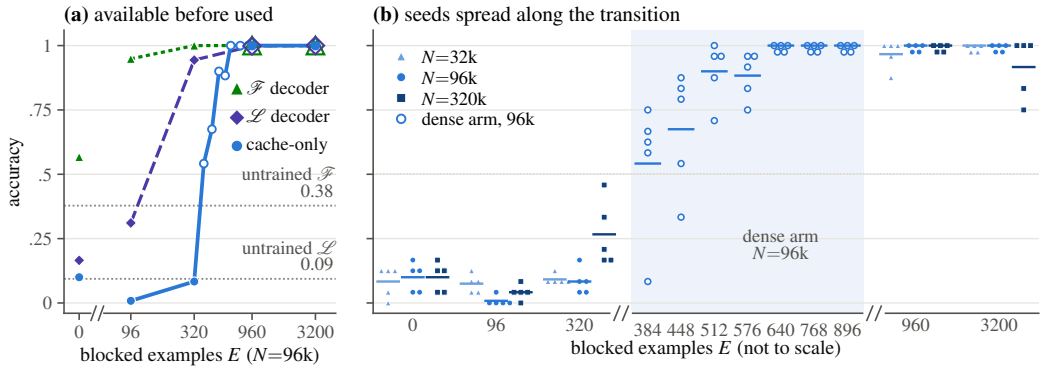


Figure 7. Three learning events, and the seed spread. (a) At N=96k, the linear availability of the verb from the whole filler span $\mathscr{F} = [23, 55)$ (triangles) and from $\mathscr{L}$ alone (diamonds), against cache-only accuracy (circles; hollow: the extra doses); marks that coincide at the ceiling are drawn as concentric rings. Dotted: the untrained models' decoders on the two spans (chance is 0.031). Availability on the cache is above its untrained floor before any causal dependence exists, which is why a probe crossing half is not an event in this geometry (Proposition 4.1). (b) Per-setting seed values behind Figure 4a, on a categorical axis that is not to scale (breaks mark the extra doses, shaded); bars are means over seeds. Tied seeds are set side by side; where five seeds tie, two sit in a second row just inside the axis, displaced by under half the 1/24 accuracy step. Inside the transition the five seeds are spread rather than split, which is P3.

D.1 Schedules

Figure 8 holds the count, the dataset size and the number of steps fixed and moves only where the 960 blocked examples fall.

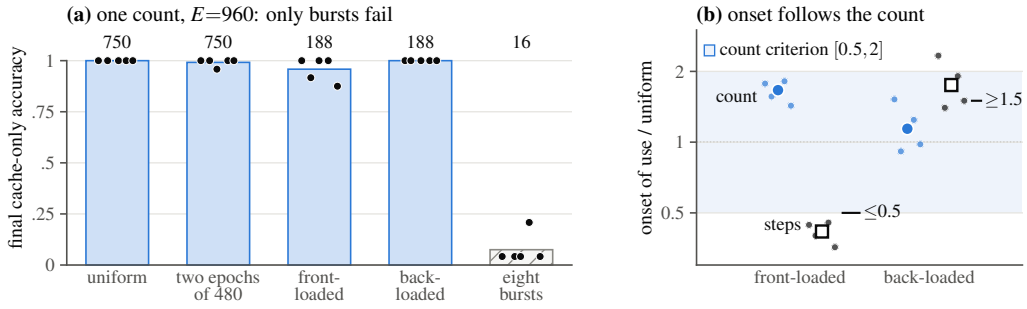


Figure 8. Count against schedule. (a) Final cache-only accuracy for the same E=960 under five schedules; dots: seeds. The number above each bar is how many of the 750 optimizer batches contain a blocked example, computed from the schedule definitions. (b) Onset of use as a ratio to the uniform schedule, in cumulative blocked count (blue circles) and in optimizer steps (black squares). Large markers: ratio of the median onsets over five seeds; small dots: per-seed ratios against the same seed's uniform run. Shaded band: $[0.5, 2]$; both count ratios lie inside it, while the step ratios lie at or beyond 0.5 (front-loaded) and 1.5 (back-loaded).

The uniform and two-epoch schedules touch all 750 optimizer batches, the front- and back-loaded ones 188, and the eight bursts 16. That puts 1.28, 5.1 and 60 blocked examples in a batch of 128 (one, four and 47 per cent), fixed by the design before any outcome was read, which is what places each schedule relative to the clipping bound (Proposition B.11). Back-loaded and two-epoch schedules match the uniform one in cache-only margin (back-loaded $+0.06$ $[-0.36, 0.53]$). The front-loaded schedule forms the cache in every seed but at a smaller margin (-3.75 $[-4.89, -2.51]$; 3.23 against 6.98), and over the 72k clean examples that follow its last blocked one it keeps answering 0.99 of items: it is short of margin, not of maintenance.

E INSIDE THE CACHE

Figure 9 resolves the cache position by position and Figure 10 by keys, values and layers (zero-training probes on the main task's checkpoints). A donor patch on the late span $\mathscr{L}$ flips no item in any model with a formed cache and at most 0.042 of items in any setting, while control patches (same answer, random, orthogonal) move the margin by less than 1.3 logits against 16.5–19.4 for a donor patch on $\mathscr{M}$. A decoder trained on $\mathscr{M}$ reads the verb on held-out field orders at 0.998 and on shifted filler at 1.0; per position it is weakest at the span's ends (0.79–0.88 for the weakest three), and untrained models' decoders sit at 0.45–0.52.

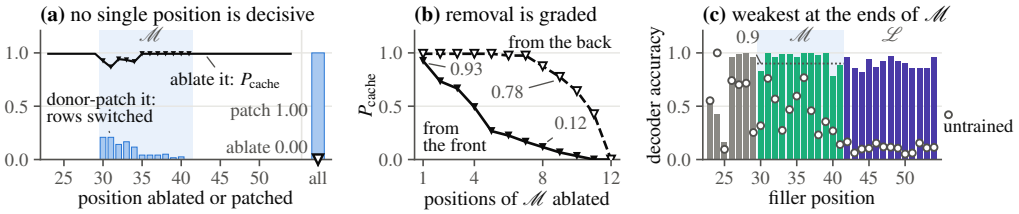


Figure 9. Inside the cache (zero-training probes at E=960, C=320; panel (c) at C=0). (a) P_{cache} after mean-ablating each single position (line, never below 0.86) and the fraction of test items flipped by donor-patching it (bars, never above 0.21); at 'all', the whole span: 0.00 and 1.00. (b) Cumulative ablations from the front (0.93 at one position, 0.12 at eight) and the back of $\mathscr{M}$ (nine can go with 0.78 left). (c) Per-position decoder accuracy, weakest at the ends (dotted: 0.9); hollow circles: untrained models.

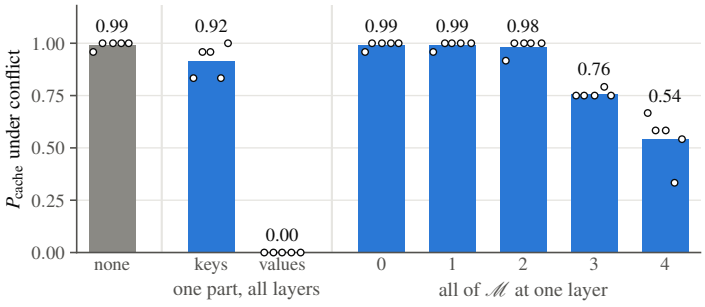


Figure 10. A value code, read by the last two layers. P_{cache} under conflict after mean-ablating $\mathscr{M}$ in the manner named: keys only, values only, or the whole span at one layer; dots: the five seeds. Ablating the keys leaves the preference at 0.92 and ablating the values destroys it; ablating any of layers 0–2 leaves it at 0.98 or above, while ablating layer 3 alone leaves 0.76 and layer 4 alone 0.54.

What the query finds at those positions therefore matters more than how much it looks at them, which is also what the conflict experiments of Section 5 report: the movement of attention mass (Appendix F) accompanies the switch, but the decision is carried on the value path.

F DETAIL OF THE SWITCH

The split of the logit, and the two coarser tests. Figure 5b attributes the answer logit at q to the key groups it reads. At E=960 and $C = 0, 96, 320, 960, 3200$ the copy's contribution under the clean readout is 8.98, 3.07, 2.63, 2.18 and 1.62 while $\mathscr{M}$'s is 1.14, 5.79, 6.94, 6.89 and 8.05; the contributions under conflict are −9.72, −3.69, −3.19, −2.63, −1.93 and 1.44, 6.98, 8.39, 8.11, 9.17. Every seed has the same sign and the seed-paired change from C=0 to C=3200 has a bootstrap interval clear of zero in each series. The cache's share, $\mathscr{M}/(\mathscr{M}+|\text{copy}|)$, is 0.113, 0.654, 0.726, 0.760, 0.832 under the clean readout and 0.128, 0.655, 0.725, 0.756, 0.826 under conflict. Table 4 sets the conflict-readout shares beside the two-route model's.

The query's attention allocation gives the same answer with less resolution. Group means match between the conditions to within 0.55 of the 40 units of mass, but individual keys move: the largest

per-key difference is 0.94 to 1.00. Attention is a coarse instrument here because a group's total can be preserved while its members are rearranged, which is why the identification of Appendix B.10 is stated on the logit decomposition instead. The third test imposes the clean readout's attention rows on the conflict forward pass; the winner survives on 0.82 to 1.0 of rows by cell, with one seed of the thirty at 0.46, so it supports the conclusion without establishing it on its own. It was run in one direction only, clean-readout rows onto conflict.

Values. At E=960, C=0 the copy wins every conflict ($P_{\text{cache}} = 0.00$, margin -11.0) although a donor patch on $\mathscr{M}$ switches every item. The flip takes 0.033–0.067% of the training set. From C=0 to 320, $\mathscr{M}$'s contribution to the conflict margin grows by 6.96 [6.52, 7.25].

Not accuracy on a subgroup. A conflict item looks like a corrupted training example, so the switch could be learned accuracy on that subgroup. Answering it would require recognizing the subgroup and weighing the routes differently there, which the equal shares on clean and conflicting inputs rule out: the behaviour generalizes to inputs on which training never separated the routes.

G THE SWITCH ACROSS DATASET SIZES AND LEARNING-RATE SCHEDULES

Three stages test P9, which an earlier form of the model of Section 4.2 already predicted. All use the main geometry with hashed fields, the uniform schedule, E=960, seeds 0–4, the same five initial states, and one pass over a prefix of the same master order. The *switch-scaling* stage moves the switch series to N=32k (250 steps, final model) and N=320k (2,500 steps, snapshots every 256 steps) at a constant learning rate of 3×10^{-4}; its 320k arm is read in a late window, the per-seed median of the last three snapshots, because a late decline was already known from the main stage. The *gradient-probe* stage trains nothing: at 255 checkpoints it computes, for each kind of example, the plain gradient of the answer loss on a fixed 256-row panel and its inner product with the gradient of the mean conflict margin and of the mean cache-only margin; the sign convention is positive toward the cache. The *learning-rate-schedule* stage retrains 65 new cells at N=96k (750 steps) and 320k (2,500 steps) with the rate held at 3×10^{-4} for 80% of the steps and decayed linearly to 3×10^{-5}, with readouts every 64 and 256 steps; it is read in the same late window. Table 10 gives P_{cache} for every arm.

Table 10. P_{cache} at E=960 by corrupted count, five-seed means. Constant rate at 96k: final models of the main and isolation stages, first item panel (pooled over both panels, P_{cache} at C=96 is 0.77, as plotted in Figure 5a); at 32k: final models; at 320k: median of the last three snapshots. Decayed rate: median of the last three snapshots. A dash: not run. $C_{1/2}$: the first linear-interpolated crossing of 0.5 by the five-seed mean (at constant rate and 96k, first panel: 55.7; Figure 1 uses the log-linear crossing of the pooled panels, 56.0).

schedule	N	0	8	16	32	64	96	160	320	960	3200	$C_{1/2}$
constant	32k	0.00	0.00	0.01	0.05	0.32	0.56	–	–	–	–	–
constant	96k	0.00	0.08	0.08	0.17	0.62	0.79	–	0.99	1.00	1.00	56
constant	320k	0.02	–	0.00	0.08	0.17	0.39	0.77	0.87	1.00	–	114
decayed	96k	0.00	–	0.01	0.33	0.23	0.84	0.92	–	–	–	78
decayed	320k	0.00	–	–	0.00	0.47	0.70	0.73	0.93	1.00	–	69

Constant rate. The brackets are $(32, 64]$ at 96k and $(64, 160]$ at 320k; at 32k no bracket exists inside the grid, since P_{cache} is 0.56 at the largest dose. A fraction would put the 32k and 320k crossings near 19 and 190. The comparison across N cannot be read at this rate: at 320k cache-only accuracy falls below 0.9 at $C \in \{0, 16\}$, because one seed's falls from 1.00 at step 2,048 to 0.13 at the last step, with no corrupted example in its stream. At $C \in \{64, 96\}$ the win rate rises with the cumulative corrupted count and then falls back while that count keeps rising: one seed at C=96 reaches 0.88 at step 1,792 and ends at 0.00.

Gradient probes. Under t-intervals the clean push below would not exclude zero (Appendix H). The positive controls hold: corrupted examples push the conflict margin toward the cache (mean effect 75.4 [69.3, 81.6]) and blocked examples push the cache-only margin before formation (45.5 [41.7, 50.4]). Late at 320k ($C \in \{64, 96\}$, steps 2,048–2,500) a clean example moves the conflict margin toward the copy (-0.049 [-0.088, -0.013]), and weighted by the counts in the stream the clean pull is -1.66 [-2.62, -0.77] times the corrupted push; at C=96 it is more negative at 320k than

at 96k (paired contrast -3.90 [-10.06, -0.47], one seed contributing -16.1). The net first-order direction of the whole stream is positive early and negative late, but both intervals include zero. On the formation side a summary statistic is taken at the checkpoints before formation (uniform schedule, N=96k, E=960, C=0; clean accuracy at least 0.95 and cache-only accuracy below 0.3): at each, the push of all clean examples on the cache-only margin divided by the push of all blocked examples, in absolute value, averaged over checkpoints within a seed and then over seeds. It is 0.014 [0.004, 0.031] (median over seeds 0.006; one seed at 0.049), below 0.25 in every seed. Because it takes absolute values it bounds the size of the clean push, not its sign; Section 4.3 reports the signed per-step values instead (Table 9), and uses this statistic nowhere. AdamW rescales each parameter by its second moment and the checkpoints carry no optimizer state, so none of these signs is guaranteed to hold for the actual update.

Decayed rate. The late decline is gone: the per-seed drop from step 1,792 to the late window at 320k, $C \in \{64, 96\}$, is -0.16 [-0.28, -0.03], against $+0.20$ [0.06, 0.37] at the constant rate. The switch is a count: the log-ratio of the two crossings is -0.13 [-0.27, -0.01], inside the count band $(-0.4, 0.4)$ and far from the fraction band $(0.8, 1.6)$; the same quantity at the constant rate is 0.72 [0.52, 0.79], in neither band. Cache-only accuracy stays at or above 0.9 in every group, and at C=0 all five seeds hold it at or above 0.98 through the late window. The 96k series is not monotone at C=64 (0.23 against 0.33 at C=32; its final-step value is 0.63, and two seeds rise from near 0 to 0.5 or more between the last two snapshots). The two schedules share the first 2,000 steps in design but not bit for bit, since GPU training is not deterministic.

H Intervals with five seeds

Every interval in the main text is a 1000-replicate percentile bootstrap. It resamples the five seeds with replacement, and contrasts are resampled as seed-paired differences; all of them reproduce exactly from the tracked aggregates. At $n = 5$ such an interval behaves like $\bar{x} \pm 1.96\, s\sqrt{4/5}/\sqrt{5}$, about 63% of the width of the Student-t interval $\bar{x} \pm 2.776\, s/\sqrt{5}$, so it undercovers; and no test on five seeds reaches two-sided $p < 0.05$, since five of five on one side gives $p = 1/16$. Table 11 therefore gives, for each interval in Sections 4 and 5, the t-interval (4 degrees of freedom, not truncated to the parameter's range); for half-crossings and their log-ratios, which are computed from the five-seed mean curve and have no per-seed value in the analysis, a leave-one-seed-out jackknife t-interval; the five seed values; and how many seeds lie on each side of the value the text compares against.

Where the two intervals disagree. Three intervals that exclude zero under the bootstrap include it under t: the clean examples' push late at 320k (negative in five of five seeds; t upper bound $+0.011$); the late decline at a constant learning rate (positive in five of five seeds, two of them by at most 0.06); and the sign of the decayed-rate log-ratio (negative in four of five seeds). We claim the first two only as seed counts and do not claim the third. The conclusions of Section 5.3 are stated on the bootstrap. Two of them also hold under t: the late decline removed (upper bound $0.04 < 0.05$) and a count across N (jackknife interval $[-0.38, 0.12]$ inside the band $(-0.4, 0.4)$). The addressing contrast holds under both intervals. The clean/blocked push ratio before formation (Appendix G) is a mean of absolute values carried by one seed (4.9% against 0.35–0.63% for the other four; median 0.6%), so its t lower bound is meaningless; it lies below 25% in every seed.

The switch bracket depends on one seed. Four of five seeds cross half between 32 and 64 corrupted examples on the first panel and on the pooled curve (on the second panel one of them crosses at 72). The fifth sits at 0.42–0.46 at C=64 and 0.29–0.33 at C=96 on both panels and crosses between 96 and 320. A bracket calls a dose above half when its bootstrap lower bound exceeds 0.5. This rule gives $(32, 64]$ on the first panel and on the pooled curve, and $(32, 96]$ on the second panel, whose interval at C=64 is $[0.45, 0.63]$. The same rule with t-intervals gives $(32, 320]$ on both panels. The switch half-crossing quoted in Figure 1 and the abstract, 56, is the log-linear crossing of the pooled mean curve (55.95); the 96k constant-rate crossing of Section 5.3 is the analysis's linear crossing of the first panel's mean curve (55.70). Both round to 56.

The half-crossings across N depend on the readout.

- *Constant rate.* The 96k reference exists only as a final model (those cells have no snapshots), while the 320k arm is read as the median of its last three snapshots. With the final snapshot at 320k as well, its crossing is 136 [128, 200] and the log-ratio is 0.90 [0.79, 1.32] rather than 0.72

[0.52, 0.79], so the bootstrap interval no longer excludes the fraction value 1.20 (its jackknife interval, [0.75, 1.04], still does). Either way the late decline leaves this comparison unreadable.

- *Decayed rate.* At 96k the win rate is still rising between the last two snapshots: at C=64 it goes from 0.04 to 0.62 in one seed and from 0.00 to 0.50 in another. With the final snapshot in place of the median of the last three, the 96k crossing is 51 [44, 58] rather than 78, and with the final snapshot on both sides the log-ratio is +0.17 [0.01, 0.44], whose upper bound leaves the count band. The late-window readout keeps it inside.

Every readout keeps the decayed-rate ratio far from the fraction value 1.20; only the late-window one puts it inside the band for a count. No readout makes the constant-rate ratio a count.

Table 11. Every interval in the main text, three ways. Boot: the seed bootstrap. t: Student-t, 4 d.f., not truncated; jk: leave-one-seed-out jackknife t for statistics of the mean curve. Seeds: seeds 0–4; per-seed half-crossings and log-ratios are computed with the analysis's own rule. Sides: seeds on each side of the reference in the last column. Pooled switch half-crossing: log-linear, pooled panels (Figure 1); constant-rate 96k half-crossing: linear, first panel (Section 5.3). Rows marked † are post hoc readout checks.

Quantity (section)	Point	Boot	t / jk	Seeds 0–4	Sides (reference)
Fixed − hashed, E=960 (4.3)	−0.63	[−0.89, −0.30]	[−1.11, −0.16]	−.04, −.79, −.46, −.96, −.92	5/5 $<$ 0
P_{cache} at C=64, panel 1 (5.2)	0.62	[0.525, 0.72]	[0.47, 0.76]	.58, .63, .79, .63, .46	4/5 $>$ .5
P_{cache} at C=64, panel 2	0.54	[0.45, 0.63]	[0.40, 0.69]	.63, .42, .67, .58, .42	3/5 $>$ .5
Switch half-crossing, pooled	56	–	–	58, 62, 53, 55, 157	4/5 in $(32, 64]$
Clean push, late 320k (5.3)	−0.049	[−0.088, −0.013]	[−0.109, 0.011]	−.015, −.089, −.112, −.005, −.024	5/5 $<$ 0
Clean pull / corrupted push	−1.66	[−2.62, −0.77]	[−3.15, −0.17]	−1.11, −2.74, −3.10, −0.28, −1.08	5/5 $<$ 0
Late drop, constant rate	0.20	[0.06, 0.37]	[−0.05, 0.44]	.50, .06, .29, .04, .08	5/5 $>$ 0
Late drop, decayed rate	−0.16	[−0.28, −0.03]	[−0.35, 0.04]	.04, −.02, −.23, −.27, −.31	4/5 $<$ 0; 5/5 $<$ .05
Half-crossing, const., 96k	56	[53, 61]	jk [49, 62]	58, 54, 52, 54, 152	4/5 in $(32, 64]$
Half-crossing, const., 320k	114	[94, 121]	jk [100, 129]	120, 109, 117, 85, 128	–
Log-ratio, constant	0.72	[0.52, 0.79]	jk [0.56, 0.88]	.72, .69, .82, .45, −.17	4/5 $>$ 0; 5/5 $<$ 1.20
† final snapshot at 320k	0.90	[0.79, 1.32]	jk [0.75, 1.04]	−.78, .89, .90, .26, −.08	5/5 $<$ 1.20
Half-crossing, decayed, 96k	78	[70, 85]	jk [67, 89]	64, 85, 71, 75, 90	–
† final snapshot	51	[44, 58]	jk [39, 62]	45, 40, 48, 53, 64	–
Half-crossing, decayed, 320k	69	[57, 79]	jk [47, 90]	53, 81, 67, 80, 69	–
Log-ratio, decayed	−0.13	[−0.27, −0.01]	jk [−0.38, 0.12]	−.18, −.04, −.06, .07, −.26	5/5 in $(-.4, .4)$; 5/5 $<$ 1.20
† final snapshot on both sides	0.17	[0.01, 0.44]	jk [−0.14, 0.49]	.17, .65, .09, .18, −.08	4/5 in $(-.4, .4)$; 5/5 $<$ 1.20