WRITTEN AS A RECORD, READ AS AN ADDRESS: WHAT A FORWARD PASS LEAVES IN AN OPERATION'S KV CACHE

Anonymous authors
Paper under double-blind review

ABSTRACT

When a language model reads an operation such as *"Swap the contents of Box F and Box B"*, its forward pass writes keys and values for those tokens into the KV cache. Prior work on entity tracking establishes what models *use*: bindings are resolved at query time rather than stored as explicit latent state. We ask what they *write* at the operation span and how it is accessed. We split a forward pass into a frozen writer and a reader: the writer's cache is recomputed without gradients, while the reader sees only the instruction and operation tokens, with all state descriptions hidden, and is trained in isolation. Anything the reader recovers was therefore already present in the unmodified cache. On a synthetic boxes task, a base reader recovers ≤ 0.07 of queried bindings against 0.75–1.00 after training, and recoverability tracks the operation's read/write footprint. We find two modes of access. Across Llama-3.1-8B and Mistral-7B, operation-span transplants causally redirect which visible state is read even when the two worlds hold identical values, revealing a *routing* record. Isolation training preserves routing and adds direct access to the *payload*, the value the operation read, from the single operand-name token in a narrow mid-depth band (layers 12–15 of 32 in Llama-3.1-8B, 14–17 in Mistral-7B) — the same site that holds the routing record. The same recipe extends to further operations, ToMi and GSM8K, but is bounded by training coverage and costs open-book accuracy. Operation tokens thus leave localized, causally recoverable records that support both routing and direct payload access, though the model that writes them reads mainly the address they carry and not the value.

1 INTRODUCTION

Language models track how a described world changes as they read: which box now holds the comb, or what a variable holds after an assignment. In a decoder-only transformer, information moves between positions only through the key/value (KV) cache, so whatever later computation knows about an update must either be stored in the cache entries written while reading it or be recomputed when a question arrives. Prior work on entity binding and tracking mostly finds the latter: bindings are resolved at query time rather than updated eagerly (Kim & Schuster, 2023; Feng & Steinhardt, 2024; Prakash et al., 2025; Oh & Demberg, 2026), and models rebuild state from visible tokens instead of maintaining it incrementally (Tang et al., 2026).

These findings concern what the model *uses*; they leave open what it *writes*. Probing alone cannot settle the question, because a probe can decode information that the model never uses (Hewitt & Liang, 2019; Elazar et al., 2021; Belinkov, 2022), and patching can show that a site matters without showing what it contains (Vig et al., 2020; Geiger et al., 2021; Zhang & Nanda, 2024). To measure the gap between what is written into the cache and what the model reads from it, we hold the writer fixed and vary only how the cache is read.

Frozen writer, trained reader. Each prompt consists of a set of state descriptions, one operation statement and a query, each occupying a contiguous span of token positions. For simplicity, we refer to these token spans as lines: the spans that state the current bindings are description lines (in code, assignments such as `a = 3`) and the span that changes them is the operation line. We mask the reader's attention to every description line and leave the operation line visible (Figure 1). This line

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.

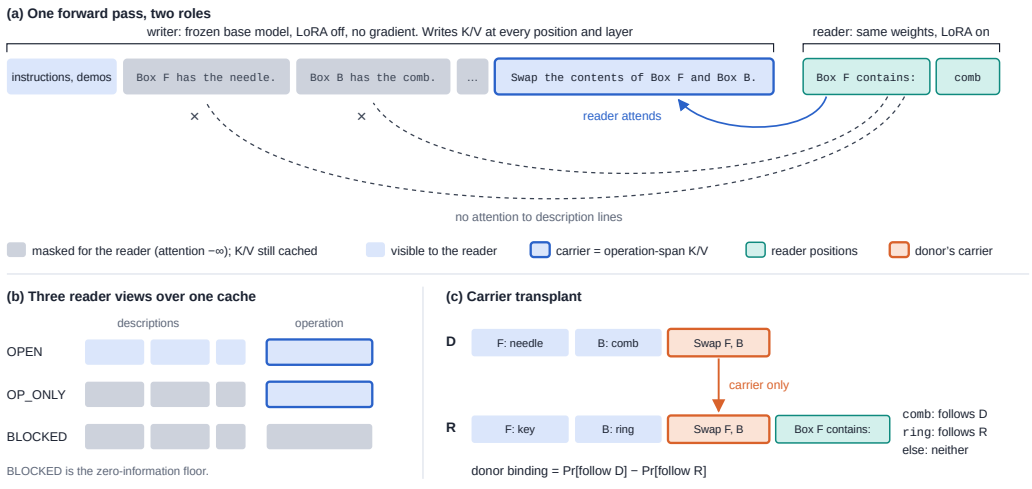

Figure 1: **Setup, shown on one boxes example**. Anything the reader answers under `OP_ONLY` must come from the operation span's K/V. Color encodes what the reader may attend to: gray spans are masked for the reader (attention logits $-\infty$; their K/V remain in the cache), blue spans are visible, and the outlined span is the carrier, i.e. the operation-span K/V. (a) The prefix is processed once by the frozen base model (writer, LoRA adapter off). The reader is the same network at the query and answer positions with the adapter on. The operation line names two box letters and no item. (b) Three views over one frozen cache. (c) A transplant copies only the carrier of a donor world D into a recipient world R that differs only in its items; the answer then follows D, follows R, or neither.

names no item, but its K/V were computed after the writer had processed the description lines, so they can depend on the items those lines mention. We then train a low-rank adapter (Hu et al., 2022) on the reader alone. The writer's weights never change and its cache is recomputed without gradients at every step, so anything the reader recovers is recoverable from the unmodified cache. This does not imply an explicit state variable or tell us whether the adapter performs a lookup or a new computation (Section 7). We study two tasks with this structure, a natural-language boxes task (Figure 1) and a code task, which lets us control exactly which variables an operation reads and which it writes. We find that the two questions come apart: the base model already uses the operation span to decide where to read, but barely recovers what it holds. Isolation training exposes the second use at the same site as the first. Our contributions are as follows:

- **Storage without native retrieval** (Section 4.1). We show that the operation span stores information that the base model rarely retrieves. With every description line hidden, the base reader answers at most 0.07 of the boxes queries on three models (Llama-3.2-1B, Llama-3.1-8B, Mistral-7B), whereas a reader trained in isolation answers 0.75–1.00 of them.
- **Operation-local footprint** (Section 4.2). We show that what can be recovered follows the operation's read/write footprint. For an assignment such as `a = b + 1`, the values of the written variable `a` and of the read-only operand `b` become recoverable from the operation span, while the values of variables the operation does not mention do not.
- **Two uses of one cache** (Section 4.3). The base model uses the carrier to select which visible line to read, even when donor and recipient hold identical values. Isolation training does not weaken this routing and adds access to operation-local values. Both results hold for three reader seeds, and both routing and direct payload replicate in Mistral-7B.
- **Where the payload is read** (Section 4.4). The trained reader reads it from the K/V of a single token, the operand name (`b` in `a = b + 1`), in layers 12–15 of 32, with a weaker contribution from 8–11 (14–17 in Mistral-7B). This is the site where the base model already keeps its routing record, so training exposes an existing record rather than creating a new store.
- **Breadth and limits** (Section 5). With each reader trained from scratch, the same recipe transfers to four non-literal code operations, ToMi and GSM8K, while ordinary fine-tuning on the same data stays at the base level; access is bounded by training coverage and costs open-book accuracy.

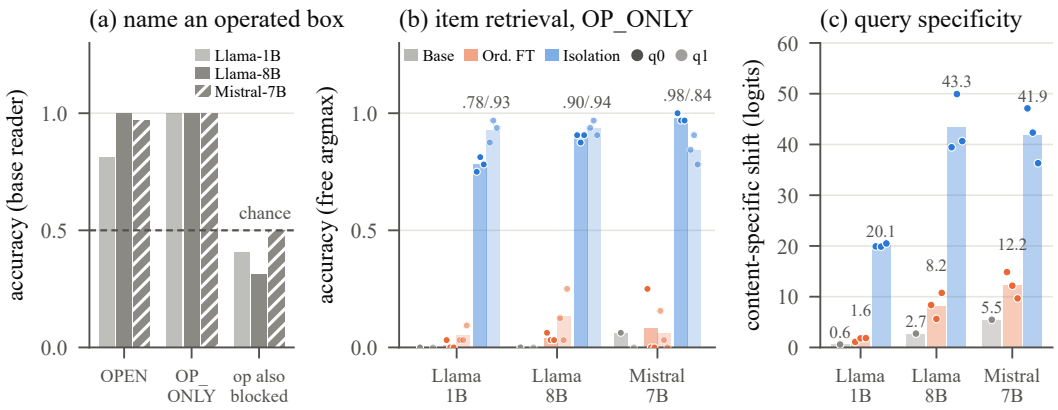

Figure 2: **Boxes task on three models** (Llama-3.2-1B, Llama-3.1-8B, Mistral-7B; 32 families each). **(a)** Mask validity: under `OP_ONLY` every base reader names an operated box perfectly, and it falls to about chance when the operation is also blocked. **(b)** Free-vocabulary retrieval of the item now in the queried box under `OP_ONLY`. Bars are means over three training seeds and dots are individual seeds; the base readers have no training seed. **(c)** Query specificity: the logit shift toward a donor's answer (relative to an irrelevant donor) when the donor changes the queried box, minus the same shift when it changes the other box. Dots are seeds.

2 METHOD

Writer, carrier, reader (Figure 1a). An instance consists of a fixed prefix, description lines s (the box's content, or the assignment in code), one operation line o and a query q. A *world* is a complete setting of s. The **writer** is the base model with the LoRA adapter disabled. It processes the prefix, s and o, and leaves K/V at every position and layer. The **carrier** is the writer's K/V at the positions of o, across all layers. We call the information causally recoverable from the carrier its **record**. The **reader** is the forward pass at q and the answer tokens. It shares all weights with the writer and differs only by an activated LoRA adapter. The adapter is applied only at the query and answer rows, so it never alters the cache it reads. We say that the cache stores information when that information is causally recoverable in this sense. No information can therefore enter the cache through the training signal; training can change only how an existing cache is read.

Views (Figure 1b). We experiment with three different views. A view sets the reader's attention logits to $-\infty$ at a set of prefix positions. `OPEN` blocks nothing. `OP_ONLY` blocks every line of the instance except o, including all description lines. `BLOCKED` blocks the whole contiguous instance, separators included, and defines the zero-information floor.

Isolation training. The reader carries a rank-16 LoRA on `q/k/v/o` and is trained with cross-entropy on the answer tokens. Its prefix cache is recomputed at every step by the base model with the adapter disabled (full recipes in Appendix A). Two arms share data, seeds, steps, optimizer and pair schedule and differ *only* in which prefix positions the reader may see during training: **isolation training** (ISO) trains under `OP_ONLY`, and **ordinary fine-tuning** (ORD) blocks nothing. ORD controls for the additional optimization and answer supervision.

Transplants (Figure 1c). A *pair* is two worlds that differ only in assigned values. Within a pair, a transplant replaces the carrier of a **recipient** world R with that of a **donor** world D at every layer, after asserting identical span positions and prefix lengths. A *matched* donor changes the value that the queried variable ends up holding. *Mismatched* and *irrelevant* donors are controls.

Metrics. Besides exact-match accuracy we report **donor binding**, $\Pr[\text{answer follows } D] - \Pr[\text{answer follows } R]$; **redirection**, the rate at which the answer follows a different recipient-visible source; and **payload readout**, the fraction of answers equal to the donor's value when every description line is masked.

3 EXPERIMENTAL SETUP

Tasks. *Boxes*: description lines are such as `Box F has the needle.`, the operation is `Swap the contents of Box F and Box B.` and the query is `Box F contains:`. Each instance carries two queries, one per operated box: q_0 asks about the box named first in the operation and q_1 about the second. We evaluate 32 item families whose box-letter sets and item sets are disjoint from the 128 training pairs (a family fixes the items and box letters from which a pair of worlds, both questions and all donors are built), and score the full-vocabulary argmax at the first answer token.

Code: four assignments with random names and values from 10–25, an operation line that contains names only (`a, b = b, a` or `a = b` or `a = b + 1`) and the query `# print(a) ->`, scored by exact match of greedy generations. The swap is queried like the boxes task, q_0 for `a` and q_1 for `b`. Unlike a swap, `a = b` writes `a` and only reads `b`, so we instead query four roles: the write target `sx`, the read-only operand `sy`, and two unmentioned variables, `sz`, which holds the same value in both worlds of a pair, and `sw`, which does not. All values are single tokens, so the two worlds of a pair are position-aligned, as a transplant requires.

Models and training. *Boxes* uses Llama-3.2-1B-Instruct (Grattafiori et al., 2024) in FP32 and, with the same pairs, schedules and seeds, Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023) in NF4 (Dettmers et al., 2023); Mistral prompts are re-rendered with its own chat template (Appendix A). Code uses Llama-3.1-8B-Instruct in NF4, because the 1B model has a low base accuracy (Appendices E and J). Mistral-7B replicates routing, payload, the payload site and the held-out footprint (Sections 4.2 to 4.4).

4 RESULTS

4.1 STORAGE WITHOUT NATIVE RETRIEVAL

All results here use `OP_ONLY`: boxes on three models (Figure 2, Table 2) and unseen code instances on the 8B model.

Base reader. Figure 2a checks the mask: it hides the description lines but not the operation, so the base model still names the operated boxes in 32/32 families and drops to about chance (0.41/0.31/0.50 on the three models) once o is also blocked. The answer item never appears in the visible text. Numbers in Appendix A and Figure 2b show that the base reader's free-vocabulary accuracy is 0.000 on Llama models, and reaches 0.062 on the Mistral model. However, the span is not ignored: a donor's carrier shifts the base model's logits toward the donor's answer in all 32 families ($S_{\text{BASE}} = 1.496$, 95% CI $[1.16, 1.86]$; Appendix B), but changes the generated answer in only 1 of 32. The base model reads the span, but not reliably enough to answer from it.

Isolation training. Figure 2b and Table 2 give the effect of training. Over three seeds, isolation training reaches 0.781 on q_0 and 0.927 on q_1, and its training loss falls from 1.90–2.00 to 0.30–0.37 (first vs. last 64 of 256 steps). Ordinary fine-tuning stays near zero on the 1B model ($q_0 \leq 0.031$, $q_1 \leq 0.094$) and at or below 0.25 in every seed of the two larger models (Table 2), although it saw the same pairs in the same order, so what matters is whether the description lines were visible during training. Figure 2c shows that the readout is query-specific: the shift toward a donor's answer (relative to an irrelevant donor) exceeds the shift caused by a donor matched to the *other* question by only 0.57 for the base model (on-target 1.60 vs. off-target 1.03), 1.09–1.84 for ORD and 19.86–20.49 for ISO (Figure 2c).

Unseen instances. We repeat the comparison on code swap with the 8B model, on 300 unseen items from an unused seed. Under `OP_ONLY` the base reader scores 0.152 [.12, .18], ordinary fine-tuning 0.112/0.120/0.142 and isolation training 0.627/0.688/0.610 over three seeds, with ordinary fine-tuning at or below the untrained reader throughout. Moreover, the effect is specific to the operation span: a later filler line, whose K/V attended to strictly more of the prompt, supports only 0.055 on validation items (Appendix C), and a donor whose values all lie outside the item reduces the trained readers to 0.013–0.040.

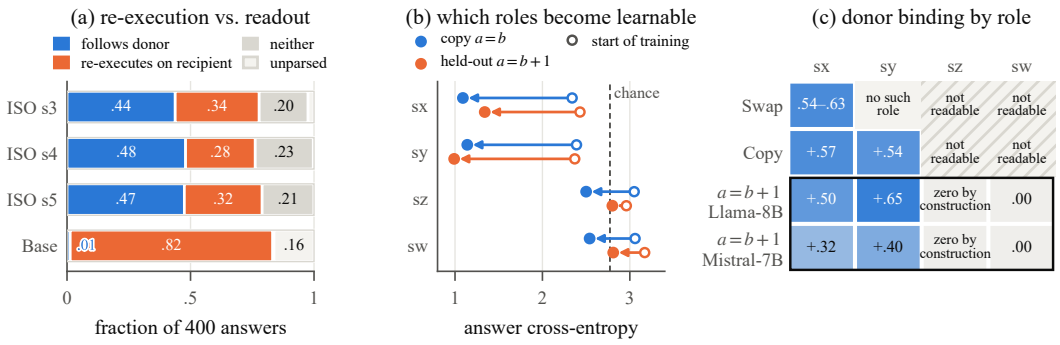

Figure 3: **Accessible information tracks the footprint (code; Llama-3.1-8B unless a row names another model). (a)** Swap under `OPEN`: the base reader re-executes on the recipient's visible state, trained readers return the donor's realized value (Appendix D). **(b)** Answer cross-entropy from the start of isolation training (open) to its end (filled); dashed line: chance $\ln 16$ (with `sx` and `sy` readable, 14 candidates remain, $\ln 14 = 2.64$). **(c)** Donor binding under `OP_ONLY`. Only the addressed roles move and bind. A swap has *no such role* at `sy`, *zero by construction* marks `sz` cells holding the same value in both worlds, and hatched cells are *not readable* by that reader.

4.2 The operation's local footprint

A box query concerns a box that the swap both reads and writes, so this section and the next two use the code task (Section 3), which separates the roles.

Re-execution versus readout. A carrier that encoded only a reusable operator could be **re-applied** to the recipient's state, and only `OPEN`, where that state remains visible, distinguishes this from a **record** of the realized result. Figure 3a shows that the base reader re-executes the operation on the recipient's visible state in 0.820 of answers and returns the donor's realized value in only 0.013: the base model uses the carrier mainly to identify *which* operation to apply. The three isolation-trained seeds instead return the donor's realized value (0.438/0.480/0.475, against 0.340/0.282/0.315 re-execution). In these worlds the donor's value is also visible on another recipient line; with disjoint values the same readers re-execute more often than they follow the donor (.108/.200 vs. .367/.408, Appendix D), so the trained reader draws on the donor's realized result without abandoning re-execution.

Learnable roles. In `a = b`, `a` is written and `b` is only read. Figure 3b tracks what isolation training makes learnable: answer cross-entropy falls for `sx` ($2.34 \to 1.09$) and `sy` ($2.39 \to 1.14$) but only slightly for `sz` and `sw` ($3.05 \to 2.50$, $3.06 \to 2.54$; chance 2.77), and under `OPEN` the trained reader answers 0.930/0.890 on the addressed roles against 0.085/0.110 on the others.

Prediction on a held-out operation. For `a = b + 1`, the footprint account implies that donor information is accessible for the **read set** $\cup$ the **write set**, here $\{b\} \cup \{a\}$. Figure 3b and the boxed rows of Figure 3c test that prediction on an operation the account was not built on: the two addressed roles became learnable ($2.43 \to 1.34$, $2.37 \to 0.99$) while `sz` and `sw` stayed at chance (2.80, 2.81), and donor binding under `OP_ONLY` was $+0.500$ for `sx` and $+0.650$ for `sy`, against 0.000 for the uninvolved frame variable `sw` (Figure 3c; `sz` holds the same value in both worlds of a pair, so its binding is zero by construction). A second reader repeats this ($+0.615$, $+0.675$, 0.000; Appendix D). On this held-out operation, accessible information follows the read/write footprint. We treat this as a functional selectivity result; it does not show that the record is complete or discrete.

4.3 Routing and payload

Following Prakash et al. (2025), we distinguish two functional interfaces. Through **routing** (addressing), the carrier determines which external source downstream computation reads. Through **payload access**, downstream computation recovers operation-local content without access to external state. We call a visible line a **causal source** of an answer if masking that line selectively removes the

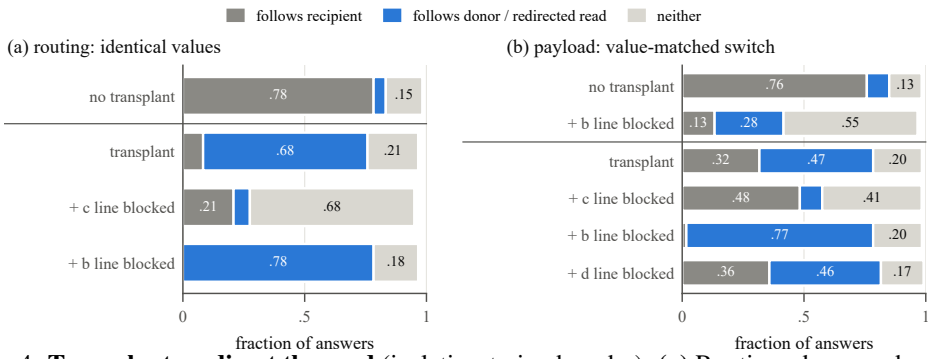

Figure 4: **Transplants redirect the read** (isolation-trained reader). **(a)** Routing: donor and recipient hold the same four values and differ only in which variable the operation reads (`a = c + 1` vs. `a = b + 1`). The read moves to the `c` line, and masking that line removes it; the untrained model behaves the same way. **(b)** Payload: the donor differs only in the operand's value (`b = 22`). After the transplant the answer moves from the `b` line to the visible line that carries `22`, and masking that line removes it.

answer. Unless stated otherwise, probes in this section use Llama-3.1-8B with the operation `TARGET = OPER + 1` over four slots (TARGET, OPER, ALT, OTH), $n = 120$. Here, TARGET and OPER are the `sx` and `sy` roles of Section 4.2; ALT is an unmentioned variable that the donor reads instead, and OTH is read by no operation in either world.

Routing. Take a recipient with `a = 14, b = 18, c = 20, d = 11` and the operation `a = b + 1` (answer `19`; OPER `b`, ALT `c`, OTH `d`). The donor holds the same four values but has the operation `a = c + 1` (a D_ROLE donor), so only the named operand differs. Figure 4a follows where the answer comes from once the donor's operation-line K/V are transplanted into the recipient: the untrained model answers `21`, the value on the `c` line plus one, on 0.642 of items against 0.008 without a transplant ($+0.633$ [+.55, +.72]), and masking `c = 20` reduces this to 0.000. The isolation-trained reader behaves the same way ($+0.625$ [+.54, +.71]; masking `c` gives 0.067; Figure 4a). The carrier therefore specifies which source to read rather than which number to output, and this routing is native. Two further reader seeds route even more strongly ($+0.942$, $+0.933$; Figure 5a), and routing replicates in Mistral-7B (base $+0.700$; trained $+0.458$ to $+0.767$). Training therefore does not trade the native interface for the learned one.

The trained reader addresses by value, not by name. Now let the recipient have `a = 14, b = 18, c = 22, d = 11` and `a = b + 1` (answer `19`), and let the donor differ only in `b = 22` (answer `23`; a D_SAME donor). Both operation lines name `b`, so a name-based address would still point to `b`, and in the recipient `22` appears only on the `c` line. Figure 4b tracks the answer as each line is masked in turn: after the transplant the three trained readers answer `23` on 0.467/0.300/0.508 of items. Masking `c = 22` changes this by -0.375 [-.47, -.29]/-0.217/-0.325 as the reader falls back to `b`, masking `b = 18` by $+0.300$/$+0.417$/$+0.342$, and masking `d = 11` by at most 0.050. If no line holds `22`, donor-following is only 0.100–0.192, and the untrained model shows none of this (0.025 in every cell). The reader thus uses the donor's value to find the visible line that carries it, so the carrier holds the value its operation read and not merely a name to re-resolve.

Direct payload. With all four description lines **masked**, the reader sees only `a = b + 1` and can output `23` only from the carrier. Figure 5b shows the trained readers do so on 0.467/0.500/0.508 of items for the D_SAME donor and on 0.408/0.467/0.400 for a D_ROLE donor that keeps the recipient's values and computes `a = c + 1` (again `23`), against at most 0.050 with no transplant (SELF) or with an unrelated donor whose values are all replaced (UNREL; Figure 5b). The carrier thus holds the value its operation read, from whichever variable, and the reader outputs it only when no description line is visible. Unlike routing, direct payload depends on training: the untrained model outputs the donor's value on only 0.117 of items. In Mistral-7B the base model already shows a weak payload (0.192), which training raises to 0.300–0.325 (Figure 5b).

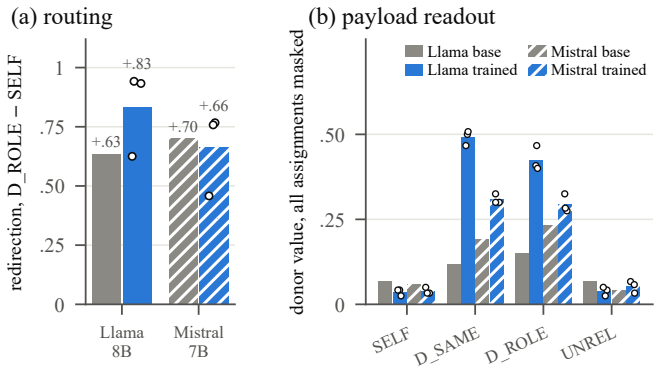

Figure 5: **Isolation training adds payload access and does not reduce native routing.** Both interfaces read the same carrier (Llama-3.1-8B and Mistral-7B, $n = 120$ per reader; trained bars: mean of three seeds, dots: seeds). SELF is the recipient's own carrier, D_SAME differs only in the operand's value, D_ROLE keeps the values but reads another variable, and UNREL replaces all values. **(a)** Routing: D_ROLE redirection minus SELF. **(b)** Donor value with all description lines masked.

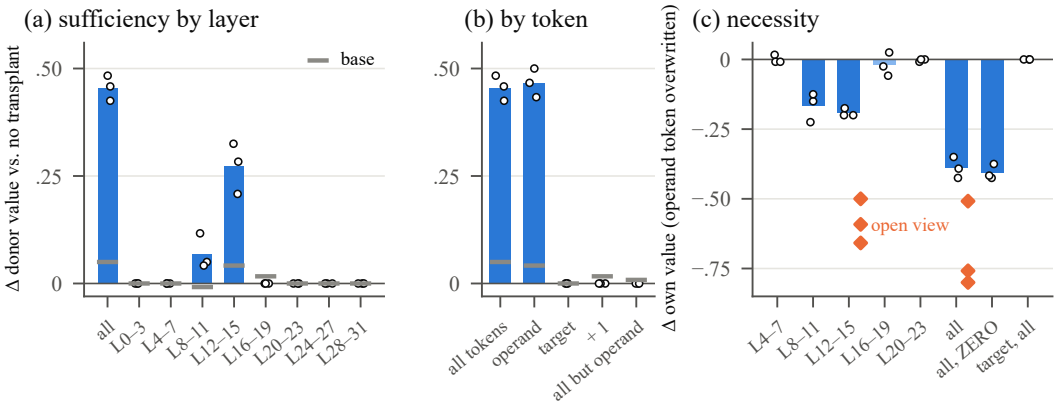

Figure 6: **The payload is read from the operand-name token in layers 12–15** (Llama-3.1-8B, all description lines masked, $n = 120$ per reader; bars: mean of three trained readers, dots: readers, gray dashes: untrained model). **(a, b)** Sufficiency: donor-value readout when a D_SAME carrier is transplanted only at one 4-layer band or only at some tokens, minus no transplant. **(c)** Necessity: change in own-value readout when the reader's operand-name K/V are overwritten (FRESH: state of a dangling name; ZERO: zeros); last bar: write-target token instead. Diamonds: description lines visible. Mistral-7B, where the site is layers 14–17: Figure 9.

The native pathway is weak, not merely unused. The base model is not ignoring the carrier—it routes with it, mainly by name and position (Appendix E)—yet it almost never answers from it. When the visible description lines are removed one at a time, with a donor whose value appears nowhere in the recipient, the trained readers climb from 0.100–0.192 to 0.475–0.500 while the base model stays at or below 0.025 until every line is masked, and reaches only 0.12 there; the decisive pair masks only the operand's line, leaving *more* text on screen than masking the other three, yet only the trained readers gain payload there, and the base model's own accuracy falls from 0.933 to 0.000 without any fallback to the carrier (Figure 8).

4.4 The payload is read from the routing record

Isolation training adds payload access (Section 4.3), but where does the payload come from? The reader could use a new representation, for instance of the result at the tokens that complete the operation, or read a value that the base model already writes for routing. The routing record is a natural candidate: in the base Llama-3.1-8B it sits at the operand-name token in layers 12–15, where transplanting it redirects the read and overwriting it cuts own execution by 0.750, and it encodes

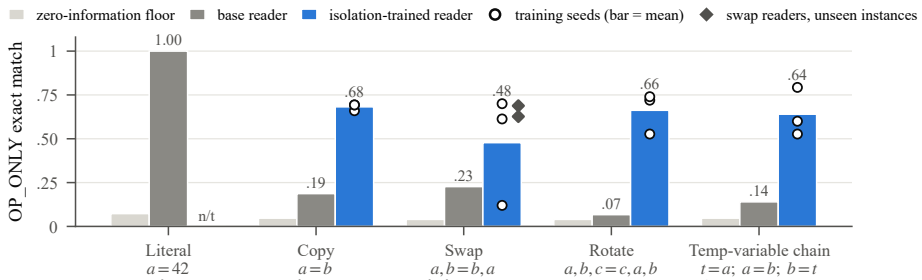

Figure 7: **Four non-literal operations and a visible-answer control** (code, 8B, `OP_ONLY`, $n = 150$). Each operation has its own freshly initialized reader. Bars are means of three training seeds (circles). Diamonds: the three swap readers of Section 4.1 on unseen instances. The literal `a = 42` puts the answer on the visible line and is not trained (n/t).

the operand's value along with its name and position (Appendix E). In all three trained readers, the payload is read from this record (Figure 6).

One token, holding the operand's value. Transplanting only the operand-name token's K/V reproduces the whole-carrier readout (0.475/0.508/0.525 vs. 0.467/0.500/0.508; no transplant $\leq$ 0.042), while every other token adds nothing. The same transplant makes the operand query return the donor's operand value (0.467–0.558), so the token carries the operand's value, and the addition is applied when the target is queried (Appendix F). Conversely, we overwrite the reader's own operand token (`b` in `a = b + 1`) with its state from a prompt whose operation names an unassigned variable instead (`a = e + 1`; FRESH). This lowers the reader's readout of its own value by 0.35–0.43, close to the floor, whereas the same overwrite of the write-target token (`a`) has no effect.

Which layers the payload is reading from. Only layers 12–15 and, more weakly, 8–11 are sufficient ($+0.21$ to $+0.33$; $+0.04$ to $+0.12$) or necessary (-0.13 to -0.23 each; at most -0.06 elsewhere). With description lines visible, the same overwrite at layers 12–15 lowers own accuracy by 0.50–0.66, so the state still serves as the address. The untrained model reads almost no payload from any part of the carrier (at most $+0.08$).

5 Breadth and Limits of Learned Access

Operations. For the four non-literal operations the base reader is weak under `OP_ONLY` (0.067–0.227). Isolation training raises accuracy to 0.527–0.793 on 11 of 12 seeds, against a floor of 0.040–0.047 on the four operations (Figure 7); the remaining swap seed reached 0.120.

ToMi. ToMi (Le et al., 2019) stories contain one move line (`Jack moved the apple to the green_box.`) and ask where the object was at the beginning, which only an earlier description line states. Treating the move as the operation line (Llama-3.1-8B, 768 steps, 200 validation questions), the base reader scores 0.040 under `OP_ONLY`, ordinary fine-tuning at most 0.040, and isolation training 1.000/0.970/0.975. Transplanted move lines bind the trained answer to the donor's start location ($\geq +0.985$) and are selective against donors that differ only in room names ($\geq +0.979$; Appendix I).

GSM8K and MMLU. On GSM8K (Cobbe et al., 2021) (Llama-3.2-1B), the writer reads the question and a reference solution, and the reader, blocked from the question and the solution's last line, must produce the final number. On the 176 items whose answer is not visible, isolation training scores 0.233/0.222/0.205, against 0.011–0.023 for ordinary fine-tuning and 0.023 for the base reader (Appendix H). The gap is not universal: on MMLU (Hendrycks et al., 2021) (Llama-3.1-8B) with the question masked, the base reader still scores 0.599 (open 0.638; Appendix I).

Limits. Learned access collapses when values move from 10–25 to 30–45 (0.030/0.040/0.168 for three seeds; Figure 10a), a failure of the answer values rather than the context values: with out-of-range context but an in-range answer accuracy holds (0.711/0.864), and with in-range context but

an out-of-range answer it collapses (0.041/0.049; crossed design, Appendix G). Training on values 10–99 restores accuracy on 30–45 (0.470/0.360/0.417) but not on 200–299 (0.015/0.025/0.000; Figure 10b), so the boundary depends at least partly on training coverage. Access does not extend to unseen operations: a reader trained on three box operations answers "nothing" on a held-out fourth in 256/256 cases.

Cost. Isolation-trained readers lose 0.105/0.168/0.017 of `OPEN` accuracy on unseen code instances and more on GSM8K (0.460–0.885 vs. 0.970; Appendix H).

6 Related work

Binding and entity tracking. Models bind entities to attributes through binding IDs (Feng & Steinhardt, 2024) and track entity states in context (Li et al., 2021; Kim & Schuster, 2023); fine-tuning strengthens existing tracking circuits rather than building new ones (Prakash et al., 2024), and Prakash et al. (2025) describe belief tracking as an address-and-payload lookback, whose vocabulary we borrow. Two studies argue that the post-operation state is not maintained: Tang et al. (2026) find that neither global nor prior states are decodable and that query-relevant information is aggregated once the query is visible, and Oh & Demberg (2026) find on the boxes swap task that a swap is a local remapping of the queried box's binding ID at readout, not a global re-encoding. We agree the base model does not retrieve that state from the operation span, and the span nonetheless carries a record a trained reader recovers.

Reading a frozen cache. Ding et al. (2026) introduce source–recipient counterfactuals similar to ours, but apply them to latent chain-of-thought checkpoints, where the reasoning is never emitted, and train nothing. Shih et al. (2026) argue on the write side that an edit counts as a state change only when a later computation uses the edited value, though their editable register comes from training the model to write its running state. KV-Skill (Han et al., 2026) trains a read interface for a frozen backbone, but on a separate residual branch and over a carrier built for the reader, as do gist tokens (Mu et al., 2023) and Patchscopes (Ghandeharioun et al., 2024). Our transplants are interchange interventions on a frozen writer's cache (Geiger et al., 2021; Meng et al., 2022; Geva et al., 2023; Zhang & Nanda, 2024). None of these combines a masking view over an unmodified cache with a frozen writer and a reader trained in isolation, which is what separates what a cache holds from what the model that wrote it retrieves.

7 Limitations

Seeds and families. Routing, direct payload and payload localization hold for three reader seeds, but their magnitudes vary (routing $+0.625$ to $+0.942$, value-matched switch 0.300–0.508, swap accuracy spread 0.58). One reader per family never learned the operand role under `OPEN`; the footprint analysis drops it, the routing and payload probes keep it, and their conclusions hold without it (Appendix D). Only Mistral-7B among five further families performed the held-out operation well enough to test, and its base model already shows a weak payload. *Reader capacity.* A rank sweep and a permuted-label control (Appendix J) bound the reader's capacity but do not separate a lookup from a learned computation over the carrier. *Scope.* We study single-step records on mostly synthetic tasks; composition across updates remains open, and ToMi and GSM8K test the access gap but not the mechanism. *Precision.* The 8B and Mistral-7B writers are NF4-quantized, and its carrier results lack a bf16 control.

8 Conclusion

We froze the model that writes a KV cache and trained only the positions that read it. In controlled state updates, the native model uses the operation span mainly to decide where to read, taking the value from visible text. Isolation training adds direct access to that value, read from the same operand-token record that serves as the address. Whether this extends to longer, composed updates in natural text remains open.

REPRODUCIBILITY STATEMENT

All masks are set differences over contiguous instance spans and are checked by assertions in the training loop. Transplants assert identical span positions and prefix lengths. Training schedules are fixed per seed before training, and the central effects were evaluated on unseen instances. Hyperparameters, seeds and full result tables are given in the appendix. Code and data generators will be released.

AI USE STATEMENT

Generative AI assistance was used for writing and debugging experiment and figure code, and, during manuscript preparation, for language editing and LaTeX diagnostics. The authors reviewed all code and are responsible for the experimental design, analyses, claims, citations, and final text.

REFERENCES

01.AI, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. Yi: Open foundation models by 01.ai. *arxiv preprint arXiv:2403.04652*, 2024.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

DeepSeek-AI: Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. Deepseek llm: Scaling open-source language models with longtermism, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023.

Yi Ding, Lijun Huang, and Menglin Yang. SCIT: Testing causal cache carriers in latent chain-of-thought models. *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? In *International Conference on Learning Representations*, 2024.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, 2021.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *International Conference on Machine Learning*, 2024.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zhaowei Han, Xiang Zhang, Bing Han, Kai Liu, Danqi Hu, and Jie Liu. KV-Skill: Forging expertise in the model's native language. *arXiv preprint arXiv:2608.05475*, 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Najoung Kim and Sebastian Schuster. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

Matthew Le, Y-Lan Boureau, and Maximilian Nickel. Revisiting the evaluation of theory of mind through question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019.

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, 2023.

Soyoung Oh and Vera Demberg. A retrieval conditioned rebinding circuit for dynamic entity tracking in large language models. *arXiv preprint arXiv:2606.08644*, 2026.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *International Conference on Learning Representations*, 2024.

Nikhil Prakash, Natalie Shapira, Arnab Sen Sharma, Christoph Riedl, Yonatan Belinkov, Tamar Rott Shaham, David Bau, and Atticus Geiger. Language models use lookbacks to track beliefs. *arXiv preprint arXiv:2505.14685*, 2025.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *arxiv preprint arXiv:2308.12950*, 2024.

Benjamin Shih, John Winnicki, and Eric Darve. When does activation steering change what a model computes from? *arXiv preprint arXiv:2606.29522*, 2026.

Zilu Tang, Qiao Zhao, Gabriel Franco, Derry Wijaya, Aaron Mueller, Sebastian Schuster, and Najoung Kim. Do language models track entities across state changes? In *International Conference on Machine Learning*, 2026. arXiv:2605.30233.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, 2020.

Qwen: An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arxiv preprint arXiv:2412.15115*, 2025.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024.

APPENDIX CONTENTS

- Appendix A: training recipes and masks
- Appendix B: native use on the boxes task
- Appendix C: controls for the operation span
- Appendix D: footprint, routing and payload: full tables
- Appendix E: the native routing record
- Appendix F: payload localization in trained readers
- Appendix G: value boundaries of learned access
- Appendix H: GSM8K
- Appendix I: ToMi and MMLU
- Appendix J: module ablation and model scale

A TRAINING RECIPES AND MASKS

	Boxes	Code
Backbone	Llama-3.2-1B-Instruct, FP32; also Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3, NF4 (bf16 compute)	Llama-3.1-8B-Instruct, NF4 (bf16 compute); carrier probes also Mistral-7B-Instruct-v0.3, NF4
Reader adapter	LoRA rank 16, $\alpha = 32$, on q/k/v/o at every layer, applied only to query and answer rows	
Optimizer	AdamW, lr 10^{-4}, weight decay 0, grad-norm clip 1.0	
Steps	256	1024 (carrier readers: 768)
One step	one pair, both worlds $\times$ both questions	one item, two queries
Scoring	first-token argmax	greedy generation, exact match
ISO / ORD	ISO blocks the whole instance except the operation line; ORD blocks nothing	

Table 1: Isolation-training recipes. Both tasks train only a reader adapter; the prefix cache is recomputed at every step by the base model with the adapter disabled, without gradient.

Masks are set differences over the whole contiguous instance, so separator newlines are blocked with everything else, and the training loop asserts that operation tokens are never blocked and description tokens always are. Every donor differs from its recipient only in assigned values, all values are single tokens (in Mistral-7B, digit tokens of equal count), and span positions and prefix lengths are asserted identical at run time (no family was skipped). Intervals are 95% bootstraps over items or families with 10,000 draws. For Mistral-7B the boxes prompts are re-rendered with the model's own chat template, with the system instruction folded into the first user turn; the first answer token is the word-initial piece of the item name, and all 32 families remain position-aligned for transplants.

Model	Reader	q0	q1	Loss (first 64 $\rightarrow$ last 64)
Llama-3.2-1B	BASE	.000	.000	–
	ORD s1 / s2 / s3	.031 / .000 / .000	.031 / .031 / .094	–
	ISO s1 / s2 / s3	.750 / .813 / .781	.875 / .969 / .938	1.90 $\rightarrow$ 0.37 / 2.00 $\rightarrow$ 0.33 / 1.98 $\rightarrow$ 0.30
Llama-3.1-8B	BASE	.000	.000	–
	ORD s1 / s2 / s3	.062 / .031 / .031	.125 / .031 / .250	–
	ISO s1 / s2 / s3	.906 / .875 / .906	.938 / .969 / .906	1.04 $\rightarrow$ 0.09 / 0.98 $\rightarrow$ 0.07 / 0.89 $\rightarrow$ 0.12
Mistral-7B	BASE	.062	.000	–
	ORD s1 / s2 / s3	.250 / .000 / .000	.156 / .031 / .000	–
	ISO s1 / s2 / s3	1.000 / .969 / .969	.844 / .906 / .781	0.34 $\rightarrow$ 0.13 / 0.61 $\rightarrow$ 0.16 / 0.51 $\rightarrow$ 0.21

Table 2: Boxes task, OP_ONLY, free-vocabulary accuracy per seed (32 families each; same pairs, schedules and seeds for all models). Loss: mean training loss over the first and last 64 of 256 steps. Mask check (base reader names an operated box under OPEN / OP_ONLY / operation also blocked): .81/1.00/.41, 1.00/1.00/.31 and .97/1.00/.50. S_{BASE}: 1.50 [1.16, 1.86], 4.24 [3.25, 5.33], 11.43 [9.49, 13.63]; a matched transplant changes the base reader's answer in 1, 0 and 3 of 32 families.

Training, validation and test splits. For the boxes task the 32 evaluation families share no box-letter set, item set or (letters, swapped pair) with the 128 training pairs; only the item vocabulary is shared, and the isolation-trained readers score .00–.06 under `BLOCKED`. For the code task, training, validation and test items come from three disjoint seeds. The prompt format, the scoring rule and the mask construction were fixed on the validation; the test set is drawn from a seed never used before and is scored by a script written in advance, with the adapters unchanged. Results are reported on the test set unless marked *validation data*.

B NATIVE USE ON THE BOXES TASK

Let $L(\cdot)$ be the logit difference between the donor world's and the recipient's answer, and $S_{\text{BASE}} = [L(\text{donor}) - L(\text{own})]_{\text{PAIR}} - [L(\text{donor}) - L(\text{own})]_{\text{IRRELEVANT}}$ on q0 (base Llama-3.2-1B, `OP_ONLY`, 32 families). $S_{\text{BASE}} = 1.496\,[1.16, 1.86]$, positive in 32/32 families. The logit shift rarely changes behavior: under `OP_ONLY` every untransplanted answer is neither the own nor the donor answer, and a matched transplant turns 1/32 into the donor answer. That family's S (2.1) lies inside the range of the others ($+0.08$ to $+4.47$), so S does not predict which families flip, and we report 1.496 and $1/32$ separately.

C CONTROLS FOR THE OPERATION SPAN

Validation data, code swap, 8B. **Null-content donor**: every variable takes a value outside the item's own four (asserted), $n = 150$ per query. **Filler span**: every prompt has a line `# state saved` after the swap, whose K/V attended to strictly more of the prompt than the swap line's; the reader sees only that line.

Reader	SELF q0/q1	NULL q0/q1	FLOOR q0/q1	Filler span only
BASE	.240 / .133	.040 / .053	.073 / .073	.060
ISO s3	.680 / .740	**.013 / .027**	.073 / .047	.055
ISO s4	.747 / .760	**.020 / .040**	.087 / .053	.055
ORD s3	.133 / .067	.040 / .020	.067 / .047	.065

Table 3: Removing the span's content drops every reader to the floor, and a later span with more context does not substitute for it.

Transplanting the recipient's own span back reproduces the untransplanted numbers exactly, so changes under other donors are caused by the replacement.

D FOOTPRINT, ROUTING AND PAYLOAD: FULL TABLES

Llama-3.1-8B unless marked Mistral-7B (both NF4), 95% bootstraps.

Reader	Donor's realized value	Re-execute on recipient	Neither	Unparsed	Donor − recompute
ISO s3	.438	.340	.198	.024	$+.098\,[+.01, +.18]$
ISO s4	.480	.282	.235	.003	$+.198\,[+.12, +.28]$
ISO s5	.475	.315	.210 (not separated)		$+.160\,[+.07, +.25]$
BASE	.013	.820	.005	.162	$-.807\,[-.85, -.77]$

Table 4: Swap under `OPEN`, matched donor, 400 answers per reader. These worlds were built by exchanging slot values, so the donor's value is visible in the recipient and the magnitudes are inflated (bottom).

Worlds ($n = 120$, one reader)	sx P(D)	sx P(R)	sy P(D)	sy P(R)
disjoint: donor value nowhere on screen	.108	.367	.200	.408
exchange: donor value on another line	.421	.267	.454	.338

Reader	Loss sx/ sy/ sz/ sw	Binding sx	Binding sy
`a = b`, Llama-3.1-8B			
s1	2.34→1.09 / 2.39→1.14 / 3.05→2.50 / 3.06→2.54	+.570 [+.49, +.65]	+.535 [+.45, +.61]
s2	2.65→1.12 / 2.54→1.13 / 3.00→2.39 / 2.92→2.53	+.55	+.53
s3	2.73→1.11 / 2.68→1.08 / 3.00→2.53 / 2.95→2.41	+.57	+.56
`a = b + 1` (held out), Llama-3.1-8B			
s1	2.43→1.34 / 2.37→0.99 / 2.96→2.80 / 3.17→2.81	+.500 [+.42, +.57]	+.650 [+.58, +.71]
s3	2.69→0.73 / 2.58→0.53 / 2.99→2.83 / 3.06→2.89	+.615	+.675
s2 (excluded)	2.66→1.02 / 2.47→0.85 / 3.13→2.76 / 3.09→2.90	(+.365)	(+.385)
`a = b + 1` (held out), Mistral-7B			
s1	0.98→0.67 / 1.03→0.55 / 1.19→1.00 / 1.20→1.11	+.315 [+.23, +.40]	+.395 [+.32, +.47]
s3	1.05→0.70 / 1.03→0.60 / 1.20→0.97 / 1.23→0.96	+.255 [+.18, +.33]	+.355 [+.28, +.42]
s2 (excluded)	1.07→0.39 / 1.07→0.40 / 1.12→1.00 / 1.20→0.99	(+.305)	(+.290)

Table 5: Footprint readers. Loss: answer cross-entropy over the first and last 64 steps per role (chance 2.77; Mistral losses sit on a digit-split scale). Binding: donor binding under `OP_ONLY`. A reader that does not reach .5 on both addressed roles with nothing masked is marked *excluded*; its bindings are given in parentheses and not used.

For $a = b + 1$, the uninvolved frame variable sw gives binding .000 (s1, s3) on Llama-3.1-8B and .000 [-.04, +.04] / +.035 [-.02, +.09] on Mistral-7B (s1 / s3; base model +.135 for sx, +.100 for sy, −.020 for sw); sz holds the same value in both worlds of a pair, so its binding is zero by construction. The read and write sets and the endpoint were written down before the s1 run. That endpoint, an exact-tuple contrast over the sx, sy and sw queries, was null (+.010 [-.03, +.05] under `OP_ONLY`): sw is uninvolved, so it carries no donor–recipient difference and the tuple cannot separate the conditions. The footprint contrast we report, addressed .588 vs. unaddressed .060 (+.527 [+.47, +.58]), was chosen afterwards.

Routing and payload probes ($n = 120$ per reader). Each world has four slots, TARGET, OPER, ALT and OTH, and the operation is `TARGET = OPER + 1`. Carriers: SELF; D_SAME (differs only in the operand value); D_ROLE (same values, reads ALT); UNREL (all values replaced). Line masks are applied at query time. These are the three footprint readers above; s2 of each family is retained here because these probes query the write target. Without s2, routing is +.625/+.933 (Llama) and +.767/+.758 (Mistral), and direct payload .467/.508 and .300/.300.

Other model families. The experiment is only meaningful in a family whose base model already performs the held-out operation: if it cannot, there is no pre-existing record for a trained reader to expose. We therefore required a base-model `OPEN` write-target accuracy above .5 ($n = 40$) before training a second family. Qwen2.5-7B (Yang et al., 2025) reaches .400, Yi-1.5-9B (01.AI et al., 2024) .500, DeepSeek-LLM-7B-Chat (Bi et al., 2024) .013 and CodeLlama-13B-Instruct (Rozière et al., 2024) .150; Mistral-7B-Instruct-v0.3 reaches .613 and was trained with three seeds (probes $n = 120$, footprint $n = 100$ families). The value-matched switch is weak: donor-following barely depends on whether the donor's value is on screen.

Probe	Trained s1 / s2 / s3	BASE
Routing, D_ROLE − SELF	+.767 [+.69, +.84] / +.458 [+.36, +.56] / +.758 [+.68, +.83]	+.700 [+.62, +.78]
Value-matched switch, donor-following (D_SAME)	.217 / .325 / .275	.008
mask ALT, change	−.067 [-.12, -.02] / −.117 [-.17, -.06] / −.025 [-.08, +.02]	−.008
mask OPER, change	+.133 [+.07, +.20] / +.058 [+.02, +.10] / +.150 [+.06, +.23]	+.017
mask OTH, change	+.075 / −.017 / .000	.000
Value absent from recipient, donor-following	.183 / .225 / .233	–
Direct payload, SELF	.033 / .033 / .050	.058
Direct payload, D_SAME	.300 / .325 / .300	.192
Direct payload, D_ROLE	.275 / .283 / .325	.233
Direct payload, UNREL	.033 / .067 / .058	.042
Direct payload, D_SAME, operand query	.325 / .308 / .308	–

Table 6: Mistral-7B, routing and payload for three trained readers (s1/s2/s3) and the base model ($n = 120$ per reader).

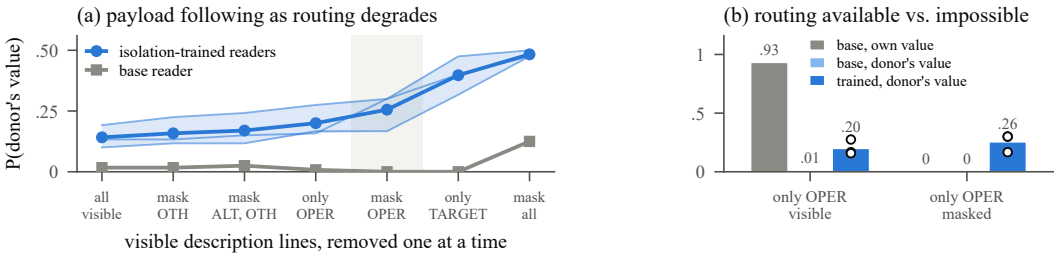

Figure 8: **The native payload pathway is weak, not merely unused** (code, Llama-3.1-8B, $n = 120$ per reader; D_SAME donor in the *absent* world, so the donor's value appears nowhere in the recipient). **(a)** Donor-value rate as visible description lines are removed one at a time; thin lines are the three trained readers, the band their range. **(b)** The shaded pair of **(a)**: masking the one line routing needs leaves *more* text visible than masking the other three, yet only the trained readers read more payload there. Circles are seeds.

E THE NATIVE ROUTING RECORD

Base models, $n = 100$. Bodies consist of six lines of the form `name = value # descriptor`. For sufficiency, the donor's target attributes (name, line position, value, descriptor) are split across recipient variables with distinct values, and each condition moves one attribute. For necessity, we overwrite the recipient's own operand-name token K/V with the state from an identical prompt that reads a dangling name (FRESH), the family mean (MEAN) or zeros (ZERO).

Moved attribute	BASE
name	+.410 [+.31, +.51]
position only	+.320 [+.23, +.41]
value	+.130 [+.07, +.20]
descriptor	−.010 [-.03, +.00]

Transplanted part	Joint	Control
whole span, all layers	+.460	+.970
layers 0–3 / 4–7 / 8–11	.000 / .000 / .000	.000
layers 12–15	+.**430** [+.34, +.53]	+.950
layers 16–23 / 24–31	.000 / .000	≤ .010
operand-name token only	+.340 [+.25, +.44]	+.830
other span tokens only	+.010	+.090

Table 7: Sufficiency on Llama-3.1-8B. Left: donor-following when one attribute is moved (whole span, all layers). Right: joint condition (all attributes compete) with partial transplants; all-attribute control in the last column. Mistral-7B: Table 8.

Transplanted part	Joint	Control
whole span, all layers	+.310 [+.22, +.40]	+.800
layers 0–3 / 4–7 / 8–11	.000 / .000 / +.010	.000
layers 12–15	+.060	+.290
layers 16–23	+.130	+.320
layers 24–31	.000	.000
layers 14–17	+.**240** [+.16, +.33]	+.750
operand-name token only	+.240 [+.16, +.33]	+.740
other span tokens only	+.020	+.030

Table 8: Sufficiency on Mistral-7B, joint condition and all-attribute control (the one-attribute conditions were not run). Band 14–17 is added because the record straddles the 4-layer grid.

Model	Band	FRESH	MEAN	ZERO
Llama-8B	0–11	.000 to −.020	.000	.000 to −.010
Llama-8B	**12–15**	−.**750** [-.83, -.66]	−.540	−.760
Llama-8B	16–31	≥ −.010	≥ −.010	≥ −.010
Mistral-7B	0–11	.000 to −.010	.000 to −.020	.000 to −.020
Mistral-7B	12–15	−.150 [-.22, -.08]	−.130	−.190
Mistral-7B	**16–19**	−.**370** [-.47, -.28]	−.260	−.330
Mistral-7B	20–31	.000	.000	.000
Mistral-7B	all 32	−.530 [-.63, -.43]	−.460	−.610
CodeLlama-13B	**12–15**	−.**400** [-.49, -.31]	−.300	−.420
CodeLlama-13B	all 40	−.760	−.650	−.730
Qwen-1.5B	16–19	−.320 [-.41, -.23]	−.240	−.360
Qwen-1.5B	all 28	−.370	−.330	−.390

Table 9: Necessity at the operand token: change in own-execution accuracy. Llama-3.1-8B (32 layers, own .990); Mistral-7B-Instruct (32 layers, own .660); CodeLlama-13B-Instruct (40 layers, own .870); Qwen2.5-1.5B-Instruct (28 layers, bf16, own .400).

Overwriting the target-name token gives .000. On Llama-8B under FRESH at 12–15, answers are the target's old value +1 in .51 of families: without the record the read defaults to the write target, and the model executes `T = T + 1`. On Mistral-7B the record straddles the 4-layer grid (layers 14–17; Table 8), overwriting the target-name token gives .000, and the tokens after the operand carry a backup copy (FRESH at all layers, −.210), as in CodeLlama-13B.

F PAYLOAD LOCALIZATION IN TRAINED READERS

The three $a = b + 1$ readers of Section 4.3 and the base model (Llama-3.1-8B), $n = 120$ per reader, all description lines masked. Sufficiency: a D_SAME donor's K/V is transplanted only at the listed layers and tokens, and we report the donor-value rate minus the no-transplant rate. Necessity: no donor; the reader's own K/V at the listed positions is overwritten, and we report the change in the own-value rate. Token groups: operand name (O), write target (T), the tokens `+ 1` (P).

	Cell	Trained s1 / s2 / s3	BASE
Sufficiency, target query	whole span, all layers	+.425 / +.458 / +.483	+.050
	operand token, all layers	+.433 / +.467 / +.500	+.042
	target token / `+ 1` / all but operand	.000 / .000 / .000	≤ +.017
	layers 8–11, all tokens	+.050 / +.042 / +.117	−.008
	layers 12–15, all tokens	+.325 / +.283 / +.208	+.042
	other six bands	.000 / .000 / .000	≤ +.017
	operand token, layers 12–15	+.300 / +.275 / +.242	+.058
Sufficiency, operand query	whole span, all layers	+.425 / +.442 / +.525	+.083
	operand token, all layers	+.425 / +.433 / +.533	+.067
	layers 12–15, all tokens	+.275 / +.375 / +.267	+.042
Necessity (intact .458 / .450 / .542)	FRESH, operand, layers 0–7	≥ −.008	
	FRESH, operand, layers 8–11	−.150 / −.125 / −.225	
	FRESH, operand, layers 12–15	−.200 / −.200 / −.175	
	FRESH, operand, layers 16–19	−.058 / −.025 / +.025	
	FRESH, operand, layers 20–31	≥ −.008	
	FRESH, operand, all layers	−.350 / −.392 / −.425	
	FRESH, operand `+1`, all layers	−.392 / −.400 / −.467	
	ZERO, operand, all layers	−.375 / −.425 / −.417	
	FRESH, target token, all layers	.000 / .000 / .000	
Necessity, `OPEN` (intact .758 / .908 / .908)	FRESH, operand, layers 12–15	−.500 / −.658 / −.592	
	FRESH, operand, all layers	−.508 / −.800 / −.758	

Table 10: Payload localization, change relative to no transplant (sufficiency) or to the intact carrier (necessity). Per-reader 95% CIs are about ±.09 for the larger cells.

Mistral-7B. The same cells for the three Mistral $a = b + 1$ readers and its base model, $n = 120$ per reader, no item skipped. Bands 14–17 and 12–19 are added because the record straddles the 4-layer grid.

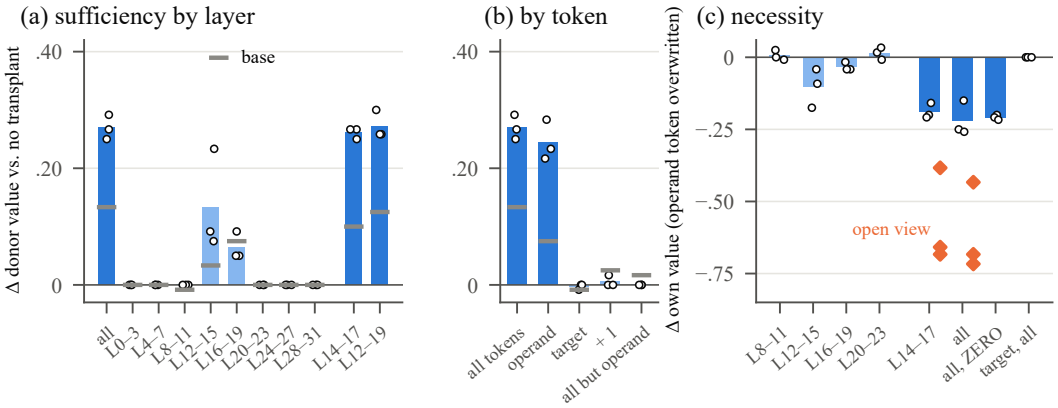

Figure 9: **Mistral-7B: the payload is read from the operand-name token in layers 14–17** (all description lines masked, $n = 120$ per reader; bars: mean of three trained readers, dots: readers, gray dashes: untrained model). Panels as in Figure 6. The 4-layer grid splits the site between bands 12–15 and 16–19, so panel (a) also shows the bands 14–17 and 12–19.

	Cell	Trained s1 / s2 / s3	BASE
Sufficiency, target query	whole span, all layers	+.267 / +.292 / +.250	+.133
	operand token, all layers	+.233 / +.283 / +.217	+.075
	target token / + 1 / all but operand	≤ +.017	≤ +.025
	layers 12–15, all tokens	+.092 / +.233 / +.075	+.033
	layers 16–19, all tokens	+.050 / +.050 / +.092	+.075
	other six bands	.000 / .000 / .000	≤ .000
	layers 14–17, all tokens	**+.250 / +.267 / +.267**	+.100
	layers 12–19, all tokens	+.258 / +.300 / +.258	+.125
	operand token, layers 14–17	+.192 / +.242 / +.225	+.067
	operand token, layers 12–19	+.208 / +.283 / +.233	+.075
Sufficiency, operand query	whole span, all layers	+.300 / +.258 / +.283	+.108
	operand token, all layers	+.308 / +.242 / +.275	+.092
	layers 14–17, all tokens	+.300 / +.250 / +.283	+.100
Necessity (intact .275 / .308 / .317)	FRESH, operand, layers 0–11	≥ −.008	
	FRESH, operand, layers 12–15	−.042 / −.175 / −.092	
	FRESH, operand, layers 16–19	−.017 / −.042 / −.042	
	FRESH, operand, layers 20–31	≥ −.008	
	FRESH, operand, layers 14–17	**−.158 / −.200 / −.208**	
	FRESH, operand, layers 12–19	−.158 / −.250 / −.233	
	FRESH, operand, all layers	−.150 / −.250 / −.258	
	FRESH, operand + + 1, all layers	−.142 / −.250 / −.275	
	ZERO, operand, all layers	−.200 / −.208 / −.217	
	FRESH, target token, all layers	.000 / .000 / .000	
Necessity, OPEN (intact .742 / .442 / .800)	FRESH, operand, layers 12–15	−.333 / −.400 / −.217	
	FRESH, operand, layers 14–17	−.658 / −.383 / −.683	
	FRESH, operand, all layers	−.683 / −.433 / −.717	

Table 11: Mistral-7B payload localization, change relative to no transplant (sufficiency) or to the intact carrier (necessity). Per-reader 95% CIs are about ±.08 for the larger cells.

G VALUE BOUNDARIES OF LEARNED ACCESS

Answer side, not context side. The value shift changes both the values in context and the value that must be output. A crossed design ($n = 400$ items) separates the two: the first letter says whether

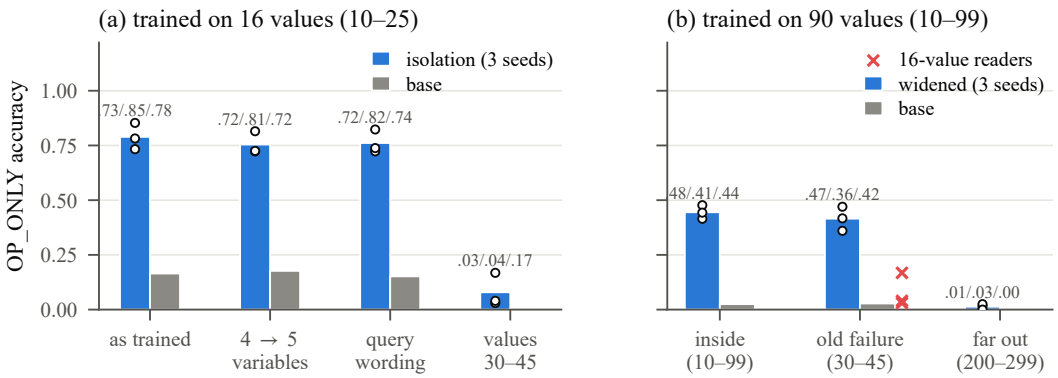

Figure 10: **Boundaries of learned access** (code swap, 8B, `OP_ONLY`, $n = 400$ pairs per set; dots are seeds). **(a)** Readers trained on 16 values (s3, s4 and the later s5) survive a change of arity and of wording but collapse on values 30–45, all of which are single tokens. **(b)** Widening the training values to 10–99 restores accuracy on the old failure set but not 200–299. Crosses: the 16-value readers on 30–45.

the other context values are in the trained range (A, 10–25) or far (F, 30–45), the second letter says the same for the answer value.

Set	BASE	ISO s3	ISO s4	ISO s5 (later)
AA	.188	.733	.851	.787
AF	.000	**.041**	**.049**	**.180**
FA	.142	.711	.864	.757
FF	.000	.024	.050	.166

Table 12: Crossed value design, `OP_ONLY`. Accuracy collapses only when the answer is out of range. For BASE, s3 and s4, `OPEN` stays at .76–.99 in every set.

Widened training. Training on `range(10, 100)` (90 values, 1024 steps) moves the loss from 4.55 to 1.44 (seed 3: 4.38 to 1.38).

H GSM8K

Llama-3.2-1B, $n = 200$. The writer reads demonstrations, the question and a reference solution. CUT_PROB_FIN blocks the question and the last reasoning line for the reader. The gold number appears in a visible intermediate line in 24 of 200 items (copyable); the remaining 176 are clean.

Reader	Clean 176	Copyable 24	All 200	`OPEN`
BASE	.023	.250	.050	.970
ORD ×3	.017/.011/.023	.250/.250/.292	.045/.040/.055	.970/1.000/1.000
ISO ×3	**.233/.222/.205**	.292/.208/.292	.240/.220/.215	.460/.575/.885

Table 13: GSM8K accuracy by stratum under CUT_PROB_FIN (first-token argmax), and `OPEN` accuracy.

Under strict free generation with exact match on clean items, ISO scores .091/.148/.125 and ORD .011/.011/.023.

I TOMI AND MMLU

ToMi. Llama-3.1-8B, same reader recipe (768 steps). We use stories from the balanced ToMi release with exactly one move line. One training step is a pair of stories whose initial containers differ (same token length) $\times$ {memory, reality} questions. Evaluation uses 200 validation memory questions. Under `OP_ONLY`, the move line's K/V are replaced by a matched donor (different initial container) or an irrelevant donor (different room names, same answer). Donor binding is P(follow donor) $-$ P(follow recipient) under the matched donor; selectivity is P(keep recipient | irrelevant) $-$ P(keep recipient | matched).

Reader	Accuracy	Donor binding	Selectivity
BASE	.040	$-.020$	$+.016$
ORD s1 / s2 / s3	.015 / .000 / .040	$-.005$ / .000 / $-.005$	$+.010$ / $+.005$ / .000
ISO s1 / s2 / s3	1.000 / .970 / .975	$+.990$ / $+.985$ / $+.990$	$+1.000$ / $+.979$ / $+.979$

Table 14: ToMi memory questions, `OP_ONLY`.

MMLU. Base Llama-3.1-8B (NF4), all 1,531 validation items, letter-restricted scoring. The question plays the role of the masked description and the answer choices that of the visible span. `OPEN` .638; question masked .599; question and choices masked .246; choices-only prompt, writer never sees the question, .346; writer reads a question from another subject, masked .299. Question masked minus other question: $+.300$ [+.27, +.33]; on the 1,002 items not answerable from the choices alone, $+.310$. The native reader already recovers question information from the choices' cache, so there is no access gap for training to close.

J MODULE ABLATION AND MODEL SCALE

Module ablation (code, 8B, `OP_ONLY`, $n = 200$, development data). Untrained LoRA modules keep `lora_B = 0` and are exactly the identity. Training only `q` and `k` reaches .705/.800 (4.2M trainable parameters), only `v` and `o` reaches .700/.335 (6.8M), and all four reaches .705/.818 (13.6M); the two halves do not sum because `k` and `v` are grouped-query projections of lower width. Both halves suffice and neither is necessary. This does not separate redirected reading from new computation.

Model scale. Under the same isolation procedure on the code swap task, Llama-3.2-1B stays at the floor, which is why the code experiments use the 8B model.

Reader capacity. Payload readout under full masking is .26/.33/.35/.50/.47 at LoRA rank 1/2/4/8/16 (routing $+.42$ to $+.66$), so readout saturates by rank 8. A control reader trained on a fixed permutation of the answer labels matches in-distribution recovery (.46 vs. .48/.39) but reaches only .15 transplant payload against .45–.50, and a reader trained on shuffled labels stays at chance.