How Instruct Models Inherit Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Instruct models increasingly read prompts written by other agents, as systems delegate simpler tasks to cheaper models. These instructions are seldom verified, but they are often the backbone of complex agentic operations. Nobody needs to verify them, as long as two assumptions are true: a model good enough at the task will not be misled by errors in the input, and if it is misled, its output will show it. We find that neither holds. Using BIG-Bench-Hard tasks rebuilt as generators, across twelve difficulty tiers and five models, we show that instruct models treat worked solutions as the answer, and therefore errors in the solution are propagated in the large majority of generations; for the hardest tiers the same holds when the error is planted mid-derivation, where no answer is stated anywhere. The rate at which errors are propagated does not fall as the model gets better at the task, and it leaves no trace in the answer. We study the copying mechanism using activation patching, and show the provided value is transported to the position that emits the answer. Hence, errors in the computation an agent passes on are not repairable by a stronger model or by reading its answers. We also study strategy prompts, in which an agent gives a task-specific overview of how the task is to be solved. A correct strategy buys about 40% of what a full solution buys, at a fraction of its cost. A strategy for the wrong task is ignored by some models and derails others, and a plausible but wrong strategy is followed at rates set by the model rather than by capability. Switching the same weights to thinking mode removes much of the copying, but less so on the hardest problems.

1 Introduction

More and more often, the prompt an instruct model reads was written by another model: a pipeline hands its partial work to the next stage, or an expensive reasoner passes its notes to a cheaper model, or an agent delegates a subtask together with what it has computed so far. This content goes mostly unchecked, and receiving agents take what it is given as a premise and reasons forward from it, inheriting whatever context they are passed (Jafarova & Alakbarli, 2026; Xie et al., 2026). Checking the content before it is read is not an option, as it amounts to solving the problem that was delegated. Hence, these agentic pipelines rest on two assumptions. The first is that a model good enough at the task will not be misled by an error in what it reads. The second is that if it is misled, we will see it in the output: the model will hesitate, retrace a step, or report the inconsistency. Rather than studying propagation in agentic pipelines, in this paper we look at the receiving step itself, and find that both assumptions fail.

What can a prompt of this kind carry? We separate its content into two kinds. A *strategy* says how the task is solved and can be reused among different instances of a problem. A *solution*, or part of one, is specific to the instance: they provide all (or part of) the computation or logical steps needed to solve the task. The two have different prices, since a strategy is written once and a solution has to be computed for every problem, and we show that a model reads them in different ways. Previous work has mostly focused on how models consume demonstrations for other problems (Min et al., 2022; Wang et al., 2023), where wrong demonstrations help as much as good ones, but in our setting of adding external strategy or solutions we find that models read correct (or wrong) solutions for its value, regardless of their capability.

To measure all of these we need access to several tasks, of varying difficulty, and we need to be able to produce good and wrong strategies or solutions. In order to do this, we rebuild three BIG-Bench-Hard tasks as generators (Suzgun et al., 2023), with twelve difficulty tiers in all, and we

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.

run five instruct models of 8B to 22B parameters on them. Every condition is measured against the same model's accuracy on the same problems with nothing added, which we call its *capability*, and we examine what happens with the accuracy under good strategies or solutions, and also ones containing errors or misleading information. Our contributions are as follows.

First we show how good strategies or solutions help models increase their accuracy, and compare them to each other. A correct strategy captures about 40% of what the full solution buys (at a fraction of its cost), with a task-dependent elbow past which more of the solution buys little (Sec. 4). A solution with one wrong value in its final state is answered with that value in a median 97% of generations, the rate does not fall with capability, and the generation is as short and as quiet as on a correct solution. Similar results also hold when the error is planted in the middle of the derivation (Sec. 5). A strategy for the wrong task can derail models, and a plausible but wrong strategy is executed at rates set by the model rather than by its capability. Further, whether the model states the strategy tells little about whether it executes it or not (Sec. 6). Activation patching and linear probes show that the planted value is transported, by a band of middle layers, to the position from which the answer is emitted, and that it is there even on the problems the model ends up answering correctly (Sec. 7). We finish the paper with a brief discussion on what changes when the same weights are switched into thinking mode: most of the copying goes away, less so as problems get harder, but the planted value is still at at the answering position (Sec. 8).

2 Related work

Corrupting provided reasoning. Perturbing a chain of thought and observing the answer is the standard instrument for asking whether this reasoning is used. Lanham et al. (2023) insert a mistake into one step and resample the remainder, and truncate chains early, to ask whether a model's own stated reasoning is post-hoc; Turpin et al. (2023) show that features of the prompt can determine the answer without being verbalized. Both ask whether a model's explanation of its own answer is faithful. We ask instead what a model does with reasoning it did not produce. One concern specific to this instrument is that a corrupted solution ending in an explicit answer statement may only test sensitivity to the position of that statement, and a recent preprint reports that deleting the statement while leaving the reasoning intact sharply reduces sensitivity to corruption at the chain's end (Garcia, 2026). Our design does not rest on that case. Every condition has its concluding sentence removed, the reference for every measurement is a question-only control, and corruption is applied throughout the solution rather than only at its end.

Finding errors, and deferring to them. Models are poor at locating the first wrong step in a chain of reasoning, although they can often correct it once told where it is (Tyen et al., 2024), and without external feedback their attempts to correct their own reasoning do not make it better (Huang et al., 2024). Models also shift towards a wrong answer that a user suggests (Sharma et al., 2024). Our receiving model is not asked to look for an error, so the question we ask comes before these works: whether it checks at all, and whether anything in its output shows it.

What is present before a model answers. Our probes say that after reading a solution the model mostly holds its last line, while the queried item becomes readable only at the answering position, after the transport. This is at odds, on the surface, with work that decodes the final answer, or the features that fix it, from the prompt representation before any token is generated (Pal et al., 2023; Yom Din et al., 2024; Dong et al., 2025), or correctness early in a reasoning trace (Zhang et al., 2025). Those tasks have an answer that is a function of a few concepts. On a compositional lookup, where the answer is one value on one line, we find no such commitment, and neither does Xu et al. (2026), who report hidden states close to random on compositional tasks until the final steps. The binding literature says why the value need not be at the prompt's end: attributes stay where they were written and are retrieved by an address (Feng & Steinhardt, 2024; Prakash et al., 2025), and a token keeps current and prior entities in separate slots of which only the current one is used at retrieval, even where the other is decodable (Bogdan & Lindsey, 2026). Our findings go one step further, since at the end of the provided solution the queried item is not decodable at all. Lastly, a compact circuit has been described for tracking which item a container holds after a swap, with no solution provided (Oh & Demberg, 2026). We conjecture that the transport we find when solutions are present takes the place of that circuit, but we did not test whether the circuit still runs in this case.

Scenario	Description	Tiers	Answer
Objects	N people hold distinct items; N pairwise swaps are applied; which item does one person hold?	$N \in \{3, 5, 7, 9, 11\}$	N options
Navigate	A sequence of turns and steps is followed from a start point; does it end where it began?	5, 10, 15, 20 instructions	Yes / No
Date	A scenario fixes today's date and asks for a date some logical steps away	1, 2, 3-4 steps	6 options

Table 1: Tasks. Three Scenarios with different tiers each, difficulty increases with the tier.

	Objects			Navigate			Date	
Model	N=3	N=7	N=11	5 steps	15 steps	20 steps	s_1	s_3
Gemma-2-9B	82.5	64.5	40.0	81.5	50.5	43.0	98.5	36.0
Llama-3.1-8B	73.0	66.5	42.5	71.0	52.5	67.5	91.0	18.5
Mistral-22B	82.5	68.0	61.5	74.0	68.0	58.5	94.0	37.5
Qwen-2.5-14B	87.0	90.5	78.0	87.0	74.5	77.5	98.5	67.5
Qwen3-14B off	100.0	99.5	93.0	90.0	98.5	94.5	98.0	73.0
Qwen3-14B on	100.0	99.5	99.0	96.0	99.5	98.0	100.0	97.0

Table 2: Capability $\mathrm{acc}_{\mathcal{M}}$of each model for a selection of tasks, see Appendix C.1 for full details. The on/off for Qwen3 indicates whether its thinking channel is open or closed. Models are ordered by capability on Objects $N = 11$, but the relative ordering of them varies trough different tasks.

3 Setup

For a task $\mathcal{T}$ we give a problem x to a model $\mathcal{M}$, and read off its answer. We call the accuracy $\mathrm{acc}_{\mathcal{M}}$ over the problems of $\mathcal{T}$ the *capability* of $\mathcal{M}$ on $\mathcal{T}$. To model an agent that hands $\mathcal{M}$ some content along with the problem, we place a prebuilt text p at the start of the assistant turn, after the user turn that holds x, and let $\mathcal{M}$ continue from it; we write this input as $x \cdot p$. We prefill p, so that the model reads it as the beginning of its own answer. This models a handoff in which the receiving model is asked to continue work that another agent has started. Accuracy on $x \cdot p$ can be higher than capability, when p is a correct strategy or holds part of a correct solution, or lower, when p is a wrong strategy or contains an error. By varying tasks and models we obtain a wide range of capabilities for which we then measure the effect of different prompts p.

Tasks. We use three scenarios from the BIG-Bench Hard dataset (Suzgun et al., 2023): Tracking Shuffled Objects (Objects), Navigate, and Date Understanding (Date), described in Table 1. We rebuild each as a generator that produces problems together with their worked solutions, since this is the only way to know every intermediate value and so to plant an error exactly where we want it. Each scenario comes in several difficulty *tiers*. Then, a *task* $\mathcal{T}$ is a scenario in a tier, and each task comprises 200 problems. The generators follow the format of the BIG-Bench Hard problems, but some of our tiers go beyond what the benchmark contains. See Appendix A for details on generation.

Models. We use five instruct models that perform no inference-time reasoning: Gemma-2-9B-it, Llama-3.1-8B-Instruct, Mistral-Small-Instruct-2409 (22B), Qwen-2.5-14B-Instruct, and Qwen3-14B with thinking switched off. They span 8B to 22B parameters and, more to the point, a wide range of capability on these tasks. In Sec. 8 we also run a reasoner, the same Qwen3-14B with thinking switched on. Table 2 gives the capability of every model (its accuracy on the problem prompt with nothing added). No model reaches 100% on the hardest tier of every scenario. Decoding follows each model's recommended settings. There is one generation per problem and condition, within a per-task token budget large enough that truncation is negligible. Generations from which no answer can be extracted are scored as wrong.

Overview of different prompts. The prompt styles used in our framework are summarized in Table 3. For *positive* prompts, meant to increase the accuracy of models, we study adding a full, synthetically-generated working solution, and different fractions of this solution. We also add a correct *strategy* for each task, which contains a basic notion on how should one solve that particular task, but is independent on the particular problem x of each task. For prompts that should have a

Style	What is placed in the assistant turn	Used in
Plain generation	nothing	throughout
Correct solution	the worked solution, in full or truncated to 25/50/75% of its length	Sec.4
Correct strategy	how to solve this task, with no values from the problem in it	Sec. 4, 7
Corrupted solution	the worked solution with one item wrong in its final state	Sec. 5, 7, 8
Corrupted mid-step	a wrong step mid-derivation, with the solution then cut, or continued correctly to a correct ending	Sec. 5, 8
Wrong strategy	the correct procedure *of a different task*	Sec. 6, 7
Deceptive strategy	a plausible but incorrect rule for the task	Sec. 6, 7
Strategy + solution	(mismatched/correct) strategy then (correct/corrupted) solution	Sec. 6

Table 3: The eight prompt styles. We try to score incorrect solutions and strategies by adoption (of the incorrect answer) whenever possible, all other prompt styles are scored by accuracy.

	Strategy	Fraction of the worked solution provided			
Task	correct strategy	25%	50%	75%	100%
Objects	0.43	0.46	0.38	0.33	0.33
Navigate	0.56	0.66	0.66	0.49	0.03
Date	0.98	0.96	0.23	0.20	0.07

Table 4: **How useful was every prompt.** Each cell is the slope of an ordinary least-squares fit of accuracy under the condition on the same model's baseline accuracy, pooled over the five instruct models and all tiers of the task. A slope of 1 means the content adds at most a constant; a slope of 0 means the content fixes the answer and the model's ability no longer matters.

negative effect, we corrupt worked solutions and / or strategies. We include a full worked solution ending in an incorrect state, and also a solution worked to the middle ending in an incorrect state. We also include one incorrect strategy, and 1-3 (depending on the task) strategies that are plausible, but still wrong; we call these *deceptive* strategies. Finally, we mix both, and prompt models with an incorrect strategy followed by a worked solution ending in an incorrect states. Every prompt string is reported in Appendix B.

4 How models use positive information

When you try to explain someone else how to solve a problem, there are two main directions. Either we describe how the task is solved, giving a more general *strategy*, or we start showing them how to work out the problem, giving them a partial *solution*. The same story happens for agents trying to raise the capability of a cheaper instruct model. So are both directions useful? which one is better?

Positive intervention helps most when there is more room. Figure 1 plots accuracy under each condition against the same model's base accuracy. Almost all tasks and models sit above the diagonal when we add a correct strategy or when we add a fraction of a worked solution. However, the vertical distance shrinks as the model's own accuracy rises; there is simply less headroom to fill.

Strategy or partial solution? To settle this question we cannot rely on gains measured in points, as these depends on the capability of each model. Instead, we fit an ordinary least-squares regression from the data in Figure 1, and also for the other quantities of a worked solution (50%, 75%, 100%). Table 4 contains the slope of each of the regressions. Here, a slope of 1 would indicate that the content we add does not affect the capability of a model at all, while a slope of 0 would indicate that the content we add suffices for the task, independently of the model's capability.

Read this way, a correct strategy does most on Objects, where the slope is 0.43, less on Navigate at 0.56, and almost nothing on Date. The intercept on Date is 4 accuracy points, meaning that the strategy does add some amount of accuracy, but it has the lowest gain. We conjecture that this is because the operations in the Date problem sets carry more variance between them, so a one-size-fits all strategy is not so realistic.

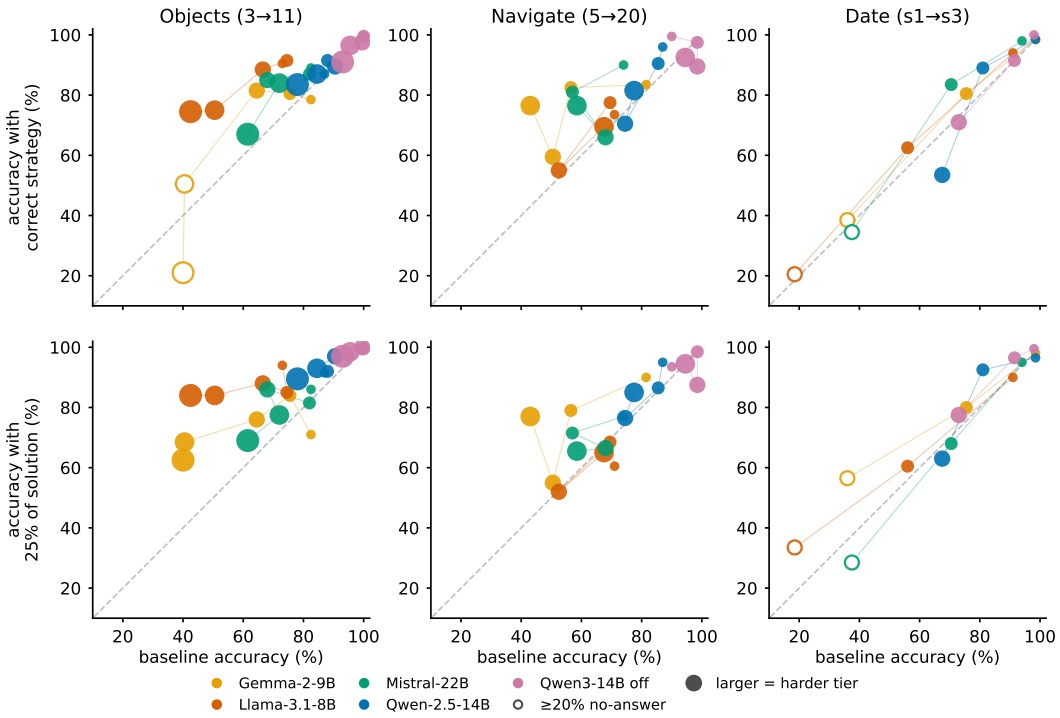


Figure 1: **Positive effects of strategy and partial worked solutions in models accuracy.** Every Task is one dot (larger dots are harder tiers within each problem), the x-axis is always the model's capability for the task, while y-axis is the accuracy when we add strategy/solution to the prompt. Hollow dots have ≥20% no-answer generations.

Comparing strategy against solution, we see that the strategy captures approximately 40% of the gains obtained by adding a full solution to the problem (Appendix C.4). But giving a full solution is infeasible in practice, so a more interesting comparison is with fragments of it. Looking again at Table 4, we see that the value of a strategy against a piece of the solution is highly dependent on the task: for Objets it provides slightly more than 25% of the solution, for Navigate more than 50%, but less than a quarter for Date. However, taking the higher cost of providing pieces of a solution, we believe that the experiments here rather recommend adding strategy as a better way of improving the capability of the model, although pieces of a solution are also helpful.

How much of a solution is needed? The slopes in Table 4 fall as more of the solution is supplied, and they fall sharply at a point that differs by task. On date the collapse is early, 0.96 at a quarter of the solution and 0.23 at a half, with little change after that, whereas on navigate the slope holds at 0.66 through half the solution, is still 0.49 at three quarters, and only collapses to 0.03 when the entire solution is present, so nothing short of the whole trace buys much. Objects falls gradually and flattens at three quarters. We do tend to see an elbow property on the slopes, so we can pick up the best cost-efficient fraction by looking at the first sharp descent in the slope.

Models tend to copy the full solution. Finally, let us look at the last column of Table 4. Under the full worked solution the slope is 0.33 on objects, 0.03 on navigate and 0.07 on date: the models own capability is almost negligible, because models tend to simply copy the solution. As we shall see in the following section, this quickly becomes a problem in cases where this solution is wrong.

5 Instruct models under negative information: Solutions

We now place in the prompt a worked solution that is wrong in one place, and ask three questions. Is the wrong value adopted? Is it adopted silently? And what, if anything, do these models check?

	Objects			Navigate			Date	
Model	N=3	N=7	N=11	5 steps	15 steps	20 steps	s_1	s_3
Gemma-2-9B	91.5	76.0	61.0	100.0	99.5	70.5	97.0	99.5
Llama-3.1-8B	93.5	97.5	92.5	–	–	–	–	–
Mistral-22B	100.0	98.0	91.0	95.0	100.0	100.0	99.5	87.5
Qwen-2.5-14B	89.0	97.0	89.0	85.0	98.0	97.5	33.5	71.0
Qwen3-14B off	76.0	98.5	98.5	99.0	100.0	99.5	98.5	98.5

Table 5: Adoption of a corrupted full solution: share of answers equal to the wrong item planted in the final state, on the tiers of Table 2. Dashes mark cells excluded for $\geq 20\%$ no-answer rate.

A wrong final value is adopted, regardless of capability. For the first experiment, the corrupted information is planted at the end of the solution, so that it points at a specific, different answer than the correct one (See Appendix B). We then measure the *adoption* of this error, how many times models output the answer implied by the corruption.

Here one would expect that the adoption would fall as the model's capability increases, but we don't see this. Table 5 gives adoption at every tier next to the model's own accuracy. The wrong value is adopted almost everywhere, with negligible measurable correlation against capability (Appendix E.1). The few exceptions we see seem to be model/task specific. Gemma-2 low adoption raises to 82%-87% for objects 9,11 if we take into account the item answered instead of the letter, as in several cases the corrupted solution appears named as the final answer, but the model assigns a different letter. The other model that appears slightly more robust against corruption is Qwen-2.5, and to some extent Qwen-3 in lower tiers.

It turns out most of the differences in adoption rate of models can be traced to their ability to recognize certain local errors, which appear to be orthogonal to the base accuracy of each model for each task. Take first Objects. On Objects our corruption gives the queried person another person's item and that person the queried person's item, with the other person drawn uniformly at random. The draw decides how the corrupted list reads. If it lands on the queried person's partner in the final swap, the corruption undoes that swap, so the last two steps of the solution are identical, and the error can be caught by comparing both lists. It turns out that Qwen-2.5 and especially Qwen-3 are able to tell this specific type of corruption apart, while the adoption rate on more complex corruption prompts is similar in all 5 models considered (Appendix E.3). For Date, the corrupted prompts that Qwen-2.5 follow a different mechanic: when the corruption is a date that is close (days apart) from the worked solution, then all models behave somewhat similar. But Qwen-2.5 is able to resist the adoption of corruptions that involve a date more than a month apart from the solution: it adopts a date one day off at 76% and a date a year off at 23%, while the other usable models adopt 94–100% at every distance. See Appendix E.4 for more details.

A natural question at this point is whether this adoption is because the corruption was planted at the end of the solution, and whether the same holds when an error is planted in a fraction of it. To answer this question, we design new set of prompts where we cut the worked solution after step k, with that step made wrong, and leave the model to finish the derivation. Since we want to measure adoption, we only analyze these conditions for Qwen3-14B, the strongest model, as it is the only one we can trust to follow up from the wrong partial solution with high accuracy. Results are in Table 6, together with two controls: one for the ability to arrive at the correct solution from the partial-but-correct solution cut at step k, and one for a full solution that introduces this error in step k but then disregards it and continues with the correct, worked out solution. Interestingly, here it does appear that capability does determine adoption, but this should be expected, as now we are forcing the model to re-derive the solution.

The adoption of errors is silent. Unfortunately, if one looks at the behavior of models on correct or corrupted full solution, there is little to tell them apart. The median generation of both answers are similar across tasks and models, and a very low percentage of the answers retrace a step of the problem or reports a problem with the provided content; the one exception is Qwen-2.5 on Date, which reports a problem in 31% of generations, and once again this traces back to how far the planted date is from the one the solution implies (Appendix E.4). Full statistics at every tier are in Appendix D.2. For the case when the error is planted at the end of a partial solution, we see some

	Objects, N			Navigate, steps			Date	
What is in the prompt	3	7	11	5	15	20	s_1	s_3
% accuracy								
cut at a correct step k (control)	100.0	100.0	97.0	100.0	88.1	92.1	99.5	100.0
cut at a wrong step k	52.5	8.5	7.5	45.9	21.8	39.6	78.0	3.0
wrong step k, then a correct ending	100.0	100.0	99.0	71.8	99.0	99.0	100.0	100.0
% adoption of error								
cut at a wrong step k	47.5	91.0	90.5	54.1	78.2	60.4	21.0	96.5

Table 6: Error in mid-derivation. Qwen3-14B with thinking off, accuracy rapidly decreases when error is planted in the middle of a worked solution, while it stays high on controls. Adoption is the percentage of prompts in which the model works out the wrong answer. Full details in Appendix D.

Model	mean damage	easiest task	hardest task	Spearman ρ
Qwen-2.5-14B	-1.1	$+0.2$	-2.4	$\mathbf{0.95}$
Gemma-2-9B	-4.1	-3.8	-4.5	$\mathbf{0.98}$
Llama-3.1-8B	-11.1	-12.7	-9.6	$\mathbf{0.74}$
Mistral-22B	-35.2	-41.3	-29.2	0.320
Qwen3-14B off	-87.2	-91.6	-82.8	0.07

Table 7: Effect of a wrong strategy to models. Here damage is the difference between the accuracy when a wrong strategy is attached and the base accuracy for each task. Spearman ρ measures the correlation between both accuracies, statistically significant values in bold.

degree of detection in the prompts based on capability, so that models would tell to report the errors in 22-68% of cases for the easy tiers, but 0-8% in the hardest tiers for Objects and Dates. When models do report, then they almost always try to re-derive the answer, when they do not report they tend to carry the error forward. For the full breakdown by tier see Appendix D.

6 Instruct models under negative information: Strategy

Full or partial solutions tend to be copied by our models even when wrong. So what about strategies? We design a new set of experiments in which each scenario is paired with 4 different strategies. One of them is completely off the mark (we actually use the strategy for a different scenario), but the three deceptive strategies are constructed so they appear plausible. For example, in Objects, one strategy tells the model that swaps cancel out, so all it needs to do is to keep track of the initial state; in Navigate one deceptive strategy tells the model that the only important part is the last step.

Wrong procedures damage by derailing models, deceptive procedures are followed. Since a wrong strategy does not lead to any type of answer, the possible damage it may cause is to confuse the model, diminishing the models accuracy for a particular problem. And indeed this is the case, see Table 7. However, the level of damage does not appear to be correlated to the capability of a model, but it does reveal a striking differences between different architectures (and even between generations of the same architecture). We also tried out mixing a wrong strategy with both a full worked solution, and a corrupt full solution, as in Section 5. Here, the presence or absence of a wrong strategy does not significantly alter accuracy scores, see Appendix F.

Next we focus on deceptive strategies. For Objects and Navigate, our deceptive strategies always lead to valid answers. When these answers are different than the correct answer, we can measure again *adoption*: in which problems the model returned the answer suggested by the deceptive strategy. Table 8 shows results for selected tasks. Once again, adoption rates do not consistently decrease with capability. On the other hand, they are very dependent on the model, and, naturally, also on the strategy. This lead us to believe models do carry over a *plausibility* check, in which a deceptive strategy would be more adopted the more plausible the strategy looks. Full details in Appendix F.

Stating a rule and executing it are different acts. We also analyze the answers to look for traces of the deceptive strategy, and compare it to the adoption rates. The two come apart in both directions,

Model	Objects				Navigate			
	swaps cancel		last swap only		last step only		forward = backward	
	N=7 n=179	N=11 n=187	N=7 n=168	N=11 n=182	15 st. n=106	20 st. n=60	15 st. n=100	20 st. n=100
Gemma-2-9B	100.0	99.5	75.6	80.2	81.1	78.3	91.0	54.0
Llama-3.1-8B	32.4	43.9	40.5	36.3	95.3	95.0	75.0	44.0
Mistral-22B	40.8	76.5	63.7	72.0	90.6	93.3	73.0	46.0
Qwen-2.5-14B	21.2	40.6	10.7	22.0	65.1	63.3	68.0	33.0
Qwen3-14B off	68.7	61.5	33.9	26.4	99.1	100.0	76.0	36.0

Table 8: Adoption rates for the most adopted deceptive strategy and the least adopted one, on selected tiers of Objects and Navigate. The value of n refers to the share of problems (out of 200) in which the answer the strategy leads to if followed is different to the correct answer.

Task	Rule	states, adopts	¬states, adopts	states, ¬adopts
Objects	swaps cancel	10%	46%	2%
	last swap only	31%	7%	24%
	parity	31%	11%	26%
Navigate	last step only	64%	11%	5%
	count turns	45%	15%	25%
	forward = backward	30%	3%	57%

Table 9: Percentage of problems where a model *states* a deceptive strategy in its answer or not, and whether it *adopts* the solution the strategy leads to, or not. Averaged over the five instruct models at the easiest and hardest tier, on problems where the strategy's answer is wrong.

see Table 9. In particular, the most damaging Objects deceptive strategy (*swaps cancel*) is adopted silently in 46% of the problems. The navigate rule *forward = backward* is the opposite, with models stating the strategy but not following it on 57% of the problems. As with corrupted solutions, this again rules out prompt inspection as a mean of catching possible errors on strategy prompts. See Appendix F for full numbers.

7 How Models Inherit Information

Results so far, where we show how instruct models adopt wrong prompts rather blindly and silently, merit a further investigation to understand how these values are propagated. Are they simply transported from the prompt to the answer, or is there computation? And what does the model hold at the moment it answers, either when answering correctly or when adopting a planted error? We answer the first question with activation patching (Meng et al., 2022; Zhang & Nanda, 2024) on Gemma-2-9B and Qwen-2.5-14B, and the second with linear probes (Alain & Bengio, 2016) on these two models and on Llama-3.1-8B and Qwen3-14B with thinking off, all on Objects at $N \in \{5, 7, 11\}$.

The answer is a transported copy of one token. We pair each correct full solution, followed by the cue "Therefore, A has the", with a corrupted copy in which the only change is the item A holds in the final state, the *planted* item; we call the token that names it the *state token*. At the cue we measure the logit difference between the correct and the planted item. We then run the corrupted prompt while overwriting the residual stream at one layer and one position with its value from the clean run, and report *restoration*, the share of the clean–corrupted difference that this recovers (0 is the corrupted run, 1 the clean run). Table 10 shows the value moving. Patching the state token at any layer up to about a third of the depth restores the correct answer fully. From the middle layers on, the state token matters less and less and the cue takes over, until at the last layer patching the cue alone restores the answer. A position in the middle of the solution restores nothing at any layer. The move is made by several heads rather than one: the best single head restores 0.17–0.24 of the gap where the full residual restores 1.00, and the top eight together reach about 0.65 (Appendix G.3).

Model holds both correct and planted answers. The other intervention we do is to fit linear probes on the residual stream at two positions: the last prompt token, which is the end of the provided

Gemma-2-9B (42 layers)				Qwen-2.5-14B (48 layers)			
layer	state token	cue	control	layer	state token	cue	control
0–12	1.00	0.00	0.00	0–13	1.00	0.00	0.00
20	0.91	0.00	0.00	25	0.87	0.00	0.00
24	0.42	0.15	0.00	30	0.54	0.24	0.00
28	0.12	0.62	0.00	35	0.19	0.60	0.00
36	0.02	0.82	0.00	40	0.05	0.83	0.00
41	0.00	1.00	0.00	47	0.00	1.00	0.00

Table 10: Restoration by layer and position, objects $N = 7$, 75 pairs per model from those that tokenize at the same length. We see how patching the state token recovers the correct answer in early layers, while patching the cue token recovers it at later stages. Other object tiers show similar behavior. The control is a position in the middle of the solution prompt.

content, and the position from which the answer is emitted. In this way, we can compare whether the linear probes can read the correct item or the item planted (by the corrupted solution or deceptive strategy). At the end of the content, the probes read the last item the solution mentions with accuracy ≥ 0.97 in every model, whether or not it belongs to the queried person, while the queried person's item is at chance. At the cue, the item the content implies is read with accuracy 0.65–0.97, both under a corrupted full solution and under a deceptive strategy; under a correct strategy, the same probe is essentially at chance. The layer at which the implied item becomes readable at the cue matches the layer at which patching the cue starts to restore the answer in Table 10.

If we restrict to the problems the model answers correctly, the planted item is still read at the cue with accuracy 0.51–0.82, and the correct item overtakes it only at about two thirds of the depth (Appendix G). Thus the planted value is carried to the answering position whether or not it is finally emitted. This essentially shows that models continue propagating the planted item even when they do not end up adopting the corrupted solution / deceptive strategy.

8 Discussion

Our results carry two main consequences. The first one is positive: reasoner agents can deliver part of their chain of thought to instruct models when they invoke them, and this tends to increase the capability of the less powerful instruct models. But this comes with a cost: errors in this solutions or ill-constructed strategies erode these gains in capability, and instruct models tend to adopt these errors blindly and silently. Hence, teams building agentic workflows should take into account that instruct models cannot be trusted to discover errors in their inherited prompts, and that it is also difficult to detect the presence of such errors from their answer. Work on self-correction finds that models are poor at locating an error in a chain of reasoning (Tyen et al., 2024; Huang et al., 2024). Our receiving models do not seem to look for one at all, beyond local checks such as an undone swap or an implausible date, and we expect checks of this kind to be part of why pipelines attenuate some errors and inherit the rest (Jamshidi et al., 2026). One limitation of our work is that we only cover the case when p is prefilled, and models are usually trained to continue their own text consistently. It could be the case that adding p as a user input may lead to lower adoption, but perhaps also to lower gains for correct strategies or solutions. This is an interesting direction for future work.

A natural question at this point is whether reasoner models exhibit similar behaviour, or what changes. We use Qwen3-14B in our experiments, which is a reasoner model, so we can compare what happens when we enable its thinking channel. In general, this causes a sharp increase in the accuracy of the problems, as now Qwen3 with thinking on drops adoption from 90% to around 40% on the higher tiers. This resistance is now modulated by capability, as adoption at the lower tiers is less than 5%. Mechanistically, here once again the planted item continues to be readable at the last answer token, but behaviorally now we see that the model actively overrides the corruption (e.g.: *wait, there seems to be a mistake...*). Surprisingly, when giving a corrupted full solution, we detect the override is in the answer, not in the thinking channel, which is, in a sense, proof that the model is partially adopting the corruption, altough we do not know if this is common for reasoner models or something specific to the Qwen3 architecture. See Appendix H for more details.

REPRODUCIBILITY STATEMENT

The generators, every prompt string (Appendix B), the answer-extraction rule and the scripts that produce every table and figure will be released and are currently avaliable at https://anonymous.4open.science/r/cot-paper-artifact-0FB3. All models are open-weight and are run with their recommended decoding settings.

LLM USAGE

In this work, we used generative AI tools for design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, and help interpreting results. We have not used generative AI tools for generating synthetic data sets or to propose or refine hypotheses, and tasks such as help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs are not applicable to this work. In addition, we used generative AI tools to create software, figures, and help organize the paper and the appendix. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Paul C. Bogdan and Jack Lindsey. Slot machines: How LLMs keep track of multiple entities. *arXiv preprint arXiv:2604.21139*, 2026.

Zhichen Dong, Zhanhui Zhou, Zhixuan Liu, Chao Yang, and Chaochao Lu. Emergent response planning in LLMs. *arXiv preprint arXiv:2502.06258*, 2025.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.17191.

Gabriel Garcia. The last word often wins: A format confound in chain-of-thought corruption studies. *arXiv preprint arXiv:2605.10799*, 2026.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024.

H. Jafarova and F. Alakbarli. Handoff hallucinations: Taxonomy, benchmark, and mitigation for multi-agent pipeline failures. In C. Zanni-Merk, E. Ismayilov, N. Ismayilova, S. Genaud, and U. Abdulkarimova (eds.), *Artificial Intelligence for Digital Transformations (AIDT 2026)*, volume 3023 of *Communications in Computer and Information Science*, pp. 325–340. Springer, 2026. doi: 10.1007/978-3-032-31319-5_22.

Saeid Jamshidi, Arghavan Moradi Dakhel, Kawser Wazed Nafi, and Foutse Khomh. Hallucination cascade: Analyzing error propagation in multi-agent LLM systems. *arXiv preprint arXiv:2606.07937*, 2026.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

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

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022. arXiv:2202.12837.

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

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C. Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, 2023. arXiv:2311.04897.

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.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, et al. Towards understanding sycophancy in language models. In *International Conference on Learning Representations*, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023. arXiv:2210.09261.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023. arXiv:2305.04388.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. arXiv:2212.10001.

Yizhe Xie, Congcong Zhu, Xinyue Zhang, Tianqing Zhu, Dayong Ye, Minfeng Qi, Huajie Chen, and Wanlei Zhou. From spark to fire: Modeling and mitigating error cascades in LLM-based multi-agent collaboration. *arXiv preprint arXiv:2603.04474*, 2026.

Liyan Xu, Mo Yu, Fandong Meng, and Jie Zhou. How far ahead do LLMs plan? uncovering the latent horizon in chain-of-thought reasoning. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. arXiv:2602.02103.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. Jump to conclusions: Short-cutting transformers with linear transformations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING)*, 2024. arXiv:2303.09435.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*, 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.

A SETUP: FULL DEFINITIONS

This appendix gives the full definition of every task, model, prompt style and measure that Section 3 summarises. The names in `typewriter` are the condition names used in the appendix tables; Table 12 maps them to the prompt styles of Table 3 in the body. Throughout the appendix, *strategy* and *procedure* are the same thing, a description of how to solve the task with no values from the problem in it; the body uses the first word and some tables the second.

A.1 TASKS

We use three synthetic task families, each modelled on a BIG-Bench-Hard task but generated by our own code, so that difficulty can be scaled, the correct answer is known by re-simulation, and a correct worked solution can be written programmatically for every problem. Each family has a difficulty knob with a small number of settings, which we call *tiers*. There are 200 problems per tier.

Objects. N people each start with a distinct item (a gift, a ball, or a soccer position; the three surface domains are rotated). N pairwise swaps are applied in order. The question asks which item one named person holds at the end, as a multiple-choice question with N options. Tiers: $N \in \{3, 5, 7, 9, 11\}$, written `obj3`–`obj11`.

Navigate. A sequence of instructions (turn left, turn right, take k steps) is followed from a start point. The question asks whether the final position is the start point, answered Yes or No. Tiers: 5, 10, 15 or 20 instructions, written `nav5`–`nav20`.

Date. A short scenario fixes today's date and asks for a date some steps away ("the day after tomorrow", "a week before the deadline"), as a multiple-choice question with six options. Tiers: one, two, or three-or-more arithmetic steps, written s_1, s_2, s_3; the s_3 problems also require a days-elapsed count.

A.2 MODELS AND DECODING

Five instruct models with no inference-time reasoning: Gemma-2-9B-it, Llama-3.1-8B-Instruct, Mistral-Small-Instruct-2409 (22B), Qwen-2.5-14B-Instruct, and Qwen3-14B with thinking switched off. One further configuration is used in Section 8: Qwen3-14B with thinking switched on (`Qwen3 on`), which shares its weights with the fifth instruct model, so the on/off comparison holds everything fixed but the mode. Table 11 lists them.

Decoding follows each model's own recommended inference settings rather than a single rule imposed across the roster. Gemma-2-9B, Llama-3.1-8B, Mistral-Small and Qwen-2.5-14B are decoded greedily. Qwen3-14B is sampled in both modes, since greedy decoding is discouraged by the model's authors, with the per-mode settings they specify: temperature 0.6 and top-p 0.95 with thinking on, temperature 0.7 and top-p 0.8 with thinking off, and top-k 20 in both. There is one generation per problem and condition, at a fixed seed ('BASE_SEED = 20260414'), within a per-task token budget large enough that budget truncation was verified to be negligible on objects. Qwen and Mistral ran with roughly twice the budget of Gemma-2 and Llama; this can only let a model finish, not change what a finished generation says. For Qwen3 with thinking on, the answer is read after the closing `</think>` tag; the 299 generations that never close the tag are read in full.

A.3 THE WORKED SOLUTION AND ITS TWO PARTS

For every problem we write a *worked solution*, programmatically, from the ground truth. On objects it lists, after every swap, who holds what. On navigate it gives the position and heading after every instruction. On date it gives one line per arithmetic step. In every condition that places a worked solution in the prompt, its final "therefore the answer is" sentence is removed, so the model has to state the conclusion itself.

We distinguish two kinds of content that a prompt can carry. *State* is content specific to the problem instance: the lists, positions and intermediate dates of the worked solution. *Procedure* is content that would be the same for every instance of the task: a description of how to solve it, with no

Model	Size	Reasoning	Role
Gemma-2-9B-it	9B	none	instruct roster
Llama-3.1-8B-Instruct	8B	none	instruct roster
Mistral-Small-Instruct-2409	22B	none	instruct roster
Qwen-2.5-14B-Instruct	14B	none	instruct roster; probes and patching
Qwen3-14B (thinking off)	14B	off	instruct roster; the intermediate-step conditions
Qwen3-14B (thinking on)	14B	on	same weights as the row above: the on/off comparison holds everything fixed but the mode

Table 11: Models. The first five are the instruct roster used throughout; the last is used only in Section 8 and in the probe tables of Appendix G.

values from the problem in it. Formally, for a task with instances q, a procedure edit is a constant, $c(q) = c_{\text{task}}$; a state edit is a function of q.

A.4 What is added to the prompt

Every prompt is the problem, followed by the line "Let's think step by step.", passed through the model's chat template. The added content of a condition is then placed at the start of the *assistant* turn, as if the model had already written it: a worked solution verbatim, or a procedure introduced by "Hint: … Following the hint, solving step by step:" (Appendix B gives every string verbatim). Table 12 lists the conditions. Conditions are grouped by what the added content is and by whether it has a *planted answer*: a specific wrong answer that a model would give if it used the content as written.

The intermediate corruption (`mid_*`) is built so that the wrong step k changes the answer: on objects it corrupts the person carrying the answer item rather than the queried person, on date it moves "today" onto the nearest other option, on navigate it miscounts one move. On navigate a miscount changes the Yes/No answer on only 85–101 of the 200 problems per tier; those are the *answer-changing* landmines used in all landmine results, and the remaining problems, where the miscount changes nothing, are reported as a null control.

A.5 Reading the answer

The answer is read from the free-text generation by a fixed rule, identical for every model. First it looks for an option letter or Yes/No in an answer phrase ("the answer is (B)", a boxed letter, a standalone letter in the closing lines). If none is found, it looks for the option's content in the last line that mentions the queried entity (the color of the gift, the full name of the position). A generation from which neither rule extracts an answer is *no-answer*; an extracted answer that is not one of the problem's options (a Yes/No on objects, a letter on navigate) is counted as wrong.

Accuracy is the share of the 200 problems answered with the correct option. We write it $\mathrm{acc}_{\mathcal{M}}(c, t)$ for model $\mathcal{M}$, condition c and tier t; `no_cot` accuracy is the model's own ability on the task, and the paper uses it throughout as the model's *capability* at that tier. Its complement, $H_{\mathcal{M}}(t) = 1 - \mathrm{acc}_{\mathcal{M}}(\text{base}, t)$, is the model's *headroom*. The *gain* of a condition is $G_{\mathcal{M}}(c; t) = \mathrm{acc}_{\mathcal{M}}(c, t) - \mathrm{acc}_{\mathcal{M}}(\text{base}, t)$; it cannot exceed the headroom.

A.6 Adoption

Accuracy cannot distinguish a model that followed wrong content from one that was confused by it, because both are simply wrong. For every condition with a planted answer we therefore compute, for each problem, the specific answer the content implies (Table 12, right column; Table 13). Problems on which that answer happens to be the correct one are *freebies* and are removed; the number that remain is n, printed in every adoption table. *Adoption* is the share of the n problems answered with the planted answer, written $A_{\mathcal{M}}(c; t)$. Every answer on those problems falls into one of four bins: the planted answer, the correct answer, *other* (some third option), and no-answer.

Name	What the prompt contains after the problem	Planted answer
Nothing, or correct state		
`no_cot`	nothing	–
`expert_stripped`	the full worked solution	–
`expert_25pct`, `50pct`, `75pct`	the first 25%, 50% or 75% of the worked solution, by length	–
`reversed_correct`	the worked solution with the order of its blocks reversed	–
Strategy only		
`correct_strategy`	a correct strategy: how to solve this task, with no values from the problem	–
`wrong_strategy`	the wrong strategy of the body: the `correct_strategy` text of a *different* task (e.g. the navigate description on an objects problem)	–
`adv_*`	a deceptive strategy: a plausible but wrong rule for this task (Table 13)	the rule's output
Corrupted state		
`corrupted_expert`	the worked solution with one item wrong in its final state: on objects the queried person is given another person's item (and vice versa); on navigate the final position and the last step are changed together; on date one step is altered	the wrong final state
`reversed_corrupted`	`corrupted_expert` with block order reversed	the wrong final state
Corrupted intermediate step (Qwen3 only)		
`mid_clean_stop`	steps $1..k$ of the worked solution, all correct, then nothing (control)	–
`mid_corrupt_stop`	steps $1..k-1$ correct, step k wrong, then nothing: the model must finish	the answer step k leads to
`mid_corrupt_full`	step k wrong, steps $k+1..N$ and the final state correct: the solution contradicts itself but ends right	the answer step k leads to
Strategy and solution together		
`wrong_strategy_correct`	`wrong_strategy` text, then the full worked solution	–
`wrong_strategy_corrupted`	`wrong_strategy` text, then `corrupted_expert`	the wrong final state

Table 12: The prompt conditions. Every condition is the problem, then the content in the middle column placed in the assistant turn (Appendix B). The right column says whether the content implies a specific wrong answer; such conditions are scored by adoption (Appendix A.6), the others by accuracy.

Task	Name	The rule, and the answer it implies
Objects	`adv_skip` (*swaps cancel*)	"the swaps tend to cancel out; just report the initial assignment" → the queried person's starting item
	`adv_last_swap` (*last swap only*)	"only the very last swap matters" → the last-swap partner's starting item if the queried person is in that swap, else their own
	`adv_parity` (*parity*)	"count how often the person appears in the swaps; even → their original item, odd → the item of the person they first swapped with" → that item ("the first partner's item" can be read as that person's starting item or the item received in that swap; the first reading is used and the second reported alongside in the appendix)
Navigate	`adv_skip` (*forward = backward*)	"you return if the number of forward steps equals the number of backward steps" → always No, because the generator never emits a backward step
	`adv_last_step` (*last step only*)	"only the last instruction matters: a step means you did not return, a turn means you did" → Yes if the last instruction is a turn
	`adv_count_turns` (*count turns*)	"an even number of turns means the direction changes cancel and you return" → Yes if the turn count is even
Date	`adv_skip`, `adv_ignore_shift`, `adv_wrong_direction`	"the first date mentioned is the answer"; "the shift is negligible, report today"; "add for 'ago', subtract for 'later'" → a date that is usually not among the options

Table 13: The deceptive strategies (`adv_*`), with the short names the body uses in parentheses. Each is a rule a model could plausibly apply to this task, and each yields a specific answer on every problem. On date, that answer is one of the six options on only 0–68% of problems, so date rules are reported by accuracy lost rather than by adoption.

The *other* bin has a natural reference: if a model guessed uniformly among the options that are neither planted nor correct, each such option would receive $1/(\#\text{options} - 2)$ of the answers. We print this chance share next to the other rate; an other rate at or below it means the answers off the planted option are scattered like guesses rather than concentrated on a third answer. On navigate there is no third option.

Answers are always scored by the extracted letter. A model can name one item in its prose and write a different letter; on objects this happens to Gemma-2 in up to 22% of answers, and re-scoring by the item named raises its adoption from 61 to 82 at $N{=}11$. This is the exception discussed in Section 5; Appendix E.2 gives the numbers.

A.7 Measures taken from the text of the generation

Four measures read the generation rather than its answer. Each uses a fixed keyword rule that the authors reviewed on a labelled sample; the agreement on that sample is given with the measure.

Detection language: the generation says the provided content is wrong ("the provided state is inconsistent", "the swap sequence gives a different result"); 42 of 45 sampled generations agreed with the rule. *States the rule*: on an `adv_*` condition, the generation restates the rule ("the swaps cancel out"); one keyword list per rule, 15 of 15 agreed on objects. *Re-derivation*: on a condition that provides the full worked solution, the generation retraces at least one step of the problem (a swap, a move, a date step) rather than only stating a conclusion; 19 of 20 agreed, and empty or one-word generations are excluded and counted. *Generation length*: the median number of characters. For `Qwen3 on` we additionally record whether the thought block between `<think>` tags is empty, and its length.

A.8 Statistics

Every rate is over 200 problems (fewer where freebies are removed), so a 95% Wilson interval is roughly ± 7 points at 50% and ± 3 at 95%; intervals are printed in the main adoption table and omitted elsewhere for legibility. Where two rates are compared within a model across problems that differ only by a random draw (Section 5), the comparison is a Mantel–Haenszel test stratified by tier. Correlations and slopes across models and tiers are descriptive summaries of 12–25 points and are not tested.

Rates for the sampled model carry sampling variance in addition to the binomial variance above, since there is one generation per problem; at 200 problems and the rates reported here the additional term is small, but Qwen3 rates should not be read as exactly reproducible.

A.9 Cells that are excluded or read with care

Four known problems affect specific cells and are marked wherever they occur. Llama-3.1-8B produces empty generations on most navigate and date problems whenever a full worked solution is in the prompt (84–98% of the time); its navigate and date cells for those conditions are printed but not used in any claim, and its objects cells are. Gemma-2's letter–item conflicts make its objects adoption an underestimate (Appendix E.2). The date `adv_*` rules imply an answer that is not among the options on most problems, so their adoption is not defined; they are reported as accuracy lost against `no_cot` on the problems where the rule is wrong, together with how often the rule is restated (Appendix F.2). And the reversed-order conditions (`reversed_*`), a worked solution with its blocks in reverse order, produce no-answer generations on most models at the larger sizes; they are not used anywhere in the paper and their tables are omitted. Cells with 20% or more no-answer generations are marked † or drawn hollow in the body. Finally, Table 14 gives `adv_parity` under the two readings of "the item of the person they first swapped with" (Table 13); the first reading is used throughout.

model	obj3	obj5	obj7	obj9	obj11
Gemma-2-9B	26.8 / 25.4	49.6 / 41.0	54.1 / 40.5	58.5 / 43.9	57.1 / 45.7
Llama-3.1-8B	26.2 / 26.9	34.8 / 33.3	29.3 / 29.8	34.1 / 30.1	35.9 / 35.7
Mistral-Small-22B	54.2 / 58.2	34.8 / 29.9	40.8 / 30.5	51.2 / 32.5	50.0 / 36.4
Qwen-2.5-14B	15.5 / 10.4	12.8 / 13.7	22.9 / 22.1	29.9 / 26.0	46.2 / 38.0
Qwen3-14B OFF	38.7 / 37.3	59.6 / 55.6	60.5 / 51.1	72.0 / 58.5	72.4 / 65.1
Qwen3-14B ON	3.6 / 2.2	3.5 / 3.4	5.7 / 6.9	19.5 / 13.0	24.4 / 16.3

Table 14: `adv_parity` adoption under the primary and alternative reading of the rule, objects.

A.10 Sanity checks on the extractor

Table 15 compares the *other* bin against its chance share at the hardest tier. Table 16 gives, per task and model, how often the letter a generation emits disagrees with the item it names, and the largest change in adoption if one re-scores by the item; Gemma-2 on objects is the one case where this matters. Table 17 is a spot-check of the extractor on 100 sampled generations, and Table 18 checks that the keyword rules never fire inside a line that merely echoes the prompt.

model	obj3	obj5	obj7	obj9	obj11	nav5	nav10	nav15	nav20	s1	s2	s3
Gemma-2-9B	6.5	8.5	12.5	16.5	23.5	0.0	0.5	0.0	0.0	0.0	1.0	1.0
Llama-3.1-8B	8.0	10.5	11.0	17.5	18.0	6.0	2.5	1.5	9.0	1.0	1.5	0.5
Mistral-Small-22B	0.0	1.0	0.5	2.0	3.5	8.0	11.5	0.0	1.0	1.0	3.0	2.5
Qwen-2.5-14B	9.0	7.5	2.0	4.0	8.0	4.0	3.0	0.0	2.0	0.5	3.5	2.5
Qwen3-14B OFF	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Qwen3-14B ON	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.5	0.0	0.0	0.5	1.5

Table 15: Other-bin rate against its chance share, hardest tier.

A.11 Freebies

Tables 19-22 shows the share of problems on which the planted content happens to imply the correct answer; these are removed before adoption is computed.

A.12 Extraction notes

The remaining tables document the extractor's edge cases: generations from which no answer could be parsed (Table 23), problems whose implied answer is not among the options (Table 24), cells

task	model	conflict % of rows	largest abs change in adoption (cell: letter → content)
date	Gemma-2-9B	1.4	2.5 (s2 `reversed_corrupted` 15.0 → 12.5)
date	Llama-3.1-8B	1.8	0.5 (s3 `reversed_corrupted` 3.0 → 2.5)
date	Mistral-Small-22B	1.9	2.5 (s2 `corrupted_expert` 93.5 → 96.0)
date	Qwen-2.5-14B	4.7	8.5 (s1 `wrong_strategy_corrupted` 30.0 → 21.5)
date	Qwen3-14B ON	0.5	0.5 (s2 `corrupted_expert` 11.5 → 11.0)
date	Qwen3-14B OFF	1.0	1.0 (s2 `mid_corrupt_stop` 33.5 → 34.5)
navigate	Gemma-2-9B	5.0	15.5 (nav5 `adv_last_step` 74.8 → 59.2)
navigate	Llama-3.1-8B	12.7	13.5 (nav5 `corrupted_expert` 51.5 → 38.0)
navigate	Mistral-Small-22B	0.7	2.5 (nav5 `reversed_corrupted` 88.5 → 91.0)
navigate	Qwen-2.5-14B	5.8	25.8 (nav10 `adv_last_step` 58.1 → 32.3)
navigate	Qwen3-14B ON	5.5	12.3 (nav15 `adv_last_step` 21.7 → 34.0)
navigate	Qwen3-14B OFF	1.6	21.4 (nav5 `adv_last_step` 64.1 → 42.7)
objects	Gemma-2-9B	5.9	21.0 (obj11 `corrupted_expert` 61.0 → 82.0)
objects	Llama-3.1-8B	4.9	18.5 (obj11 `wrong_strategy_corrupted` 75.0 → 93.5)
objects	Mistral-Small-22B	1.4	2.8 (obj7 `adv_skip` 40.8 → 43.6)
objects	Qwen-2.5-14B	4.5	10.0 (obj11 `wrong_strategy_corrupted` 85.5 → 95.5)
objects	Qwen3-14B ON	0.6	5.0 (obj9 `adv_skip` 1.7 → 6.7)
objects	Qwen3-14B OFF	0.4	3.0 (obj7 `adv_last_swap` 33.9 → 31.0)

Table 16: Letter–item conflicts: adoption by the letter emitted vs by the item named.

task	bin	rows	fine	extractor artifact	pipeline error	implied suspect
objects	gold	10	10	0	0	0
objects	implied	10	10	0	0	0
objects	other	10	6	4	0	0
objects	na	10	10	0	0	0
navigate	gold / implied / na	30	30	0	0	0
navigate	other	0 (none exist outside invalid letters)	—	—	—	—
date	gold	10	10	0	0	0
date	implied	10	10	0	0	0
date	other	10	7	3	0	0
date	na	10	10	0	0	0

Table 17: Spot-check review of the extractor.

task	model	gens with echo lines %	keyword hits inside echo %
date	Gemma-2-9B	0.0	0.0
date	Llama-3.1-8B	0.0	0.0
date	Mistral-Small-22B	0.2	0.0
date	Qwen-2.5-14B	0.5	0.0
date	Qwen3-14B OFF	0.2	0.0
date	Qwen3-14B ON	10.0	0.0
navigate	Gemma-2-9B	0.0	0.0
navigate	Llama-3.1-8B	0.0	0.0
navigate	Mistral-Small-22B	0.0	0.0
navigate	Qwen-2.5-14B	0.4	0.0
navigate	Qwen3-14B OFF	0.0	0.0
navigate	Qwen3-14B ON	36.1	0.0
objects	Gemma-2-9B	2.6	0.0
objects	Llama-3.1-8B	1.4	0.0
objects	Mistral-Small-22B	1.9	0.0
objects	Qwen-2.5-14B	1.6	0.0
objects	Qwen3-14B OFF	58.9	0.0
objects	Qwen3-14B ON	86.0	0.0

Table 18: Echo exclusion: how often generations restate the prompt's own wording.

condition	obj3	obj5	obj7	obj9	obj11
corrupted_expert	0.0	0.0	0.0	0.0	0.0
wrong_strategy_corrupted	0.0	0.0	0.0	0.0	0.0
mid_corrupt_stop	0.0	0.0	0.0	0.0	0.0
mid_corrupt_full	0.0	0.0	0.0	0.0	0.0
adv_skip	34.0	14.5	10.5	10.0	6.5
adv_last_swap	0.0	16.0	16.0	14.0	9.0
adv_parity	16.0	29.5	21.5	18.0	22.0
reversed_corrupted	0.0	0.0	0.0	0.0	0.0

Table 19: Freebie fractions, objects.

condition	nav5	nav10	nav15	nav20
corrupted_expert	0.0	0.0	0.0	0.0
wrong_strategy_corrupted	0.0	0.0	0.0	0.0
mid_corrupt_stop	53.8	50.3	49.5	49.5
mid_corrupt_full	53.8	50.3	49.5	49.5
adv_skip	58.0	50.5	50.0	50.0
adv_last_step	48.5	69.0	47.0	70.0
adv_count_turns	23.5	76.0	22.5	74.0
reversed_corrupted	0.0	0.0	0.0	0.0

Table 20: Freebie fractions, navigate.

condition	s1	s2	s3
corrupted_expert	0.0	0.0	0.0
wrong_strategy_corrupted	0.0	0.0	0.0
mid_corrupt_stop	0.0	0.0	0.0
mid_corrupt_full	0.0	0.0	0.0
reversed_corrupted	0.0	0.0	0.0

Table 21: Freebie fractions, date.

tier	condition	freebie %
s1	adv_skip	31.0
s1	adv_wrong_direction	71.5
s2	adv_skip	8.0
s2	adv_wrong_direction	39.0
s3	adv_wrong_direction	26.0
obj11	adv_parity	35.5
obj3	adv_parity	33.0
obj5	adv_parity	41.5
obj7	adv_parity	34.5
obj9	adv_parity	38.5

Table 22: Freebie fractions under the alternative oracle readings.

with fewer than 190 usable problems (Table 25), extractions that are not a valid option (Table 26), and no-answer rows that are empty or a bare mark (Table 27).

benchmark	condition	reason	rows
synthnavigate10	mid_corrupt_full	implied_underivable	1
synthnavigate10	mid_corrupt_stop	implied_underivable	1
synthnavigate5	mid_corrupt_full	implied_underivable	16
synthnavigate5	mid_corrupt_stop	implied_underivable	16

Table 23: Parse failures.

benchmark	condition	not in options	n_used	%
synthdate_s1	adv_skip	104	152	68.4
synthdate_s1	adv_ignore_shift	65	92	70.7
synthdate_s1	adv_wrong_direction	80	92	87.0
synthdate_s1	mid_corrupt_stop	1	200	0.5
synthdate_s1	mid_corrupt_full	1	200	0.5
synthdate_s2	adv_skip	139	184	75.5
synthdate_s2	adv_ignore_shift	134	200	67.0
synthdate_s2	adv_wrong_direction	163	200	81.5
synthdate_s2	mid_corrupt_stop	3	200	1.5
synthdate_s2	mid_corrupt_full	3	200	1.5
synthdate_s3	adv_skip	200	200	100.0
synthdate_s3	adv_ignore_shift	151	200	75.5
synthdate_s3	adv_wrong_direction	174	200	87.0
synthdate_s3	mid_corrupt_stop	1	200	0.5
synthdate_s3	mid_corrupt_full	1	200	0.5

Table 24: Problems whose implied answer is not among the options.

task	condition	tiers
date	adv_ignore_shift	s1
date	adv_skip	s1, s2
date	adv_wrong_direction	s1
navigate	adv_count_turns	nav10, nav15, nav20, nav5
navigate	adv_last_step	nav10, nav15, nav20, nav5
navigate	adv_skip	nav10, nav15, nav20, nav5
navigate	mid_corrupt_full	nav10, nav15, nav20, nav5
navigate	mid_corrupt_stop	nav10, nav15, nav20, nav5
objects	adv_last_swap	obj11, obj5, obj7, obj9
objects	adv_parity	obj11, obj3, obj5, obj7, obj9
objects	adv_skip	obj11, obj3, obj5, obj7, obj9

Table 25: Cells with fewer than 190 usable problems.

task	model	rows
date	Gemma-2-9B	123
date	Llama-3.1-8B	70
date	Mistral-Small-22B	177
date	Qwen-2.5-14B	19
date	Qwen3-14B OFF	137
date	Qwen3-14B ON	31
navigate	Mistral-Small-22B	196
navigate	Qwen-2.5-14B	1
navigate	Qwen3-14B OFF	28
navigate	Qwen3-14B ON	77
objects	Llama-3.1-8B	117
objects	Mistral-Small-22B	92
objects	Qwen-2.5-14B	5
objects	Qwen3-14B OFF	932

Table 26: Extractions that are not a valid option.

task	model	rows
date	Gemma-2-9B	928
date	Llama-3.1-8B	2827
date	Mistral-Small-22B	155
date	Qwen3-14B OFF	209
date	Qwen3-14B ON	3
navigate	Llama-3.1-8B	3442
objects	Gemma-2-9B	551
objects	Llama-3.1-8B	1962

Table 27: No-answer rows that are empty or a bare mark.

B PROMPTS AS SHOWN TO THE MODEL

Every string in this appendix is copied verbatim from the files the generation runs read; nothing is paraphrased. Arrows and dashes are rendered in ASCII.

B.1 HOW A PROMPT IS ASSEMBLED

The model never sees a bare question. Three things are put together.

The user message. The question is followed by a fixed line, the same for every task:

```
{question}
Let's think step by step.
```

The chat template. That user message goes through the model's own chat template, with the assistant turn opened. For Qwen-2.5-14B-Instruct the result is:

```
<|im_start|>system
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>
<|im_start|>user
{user message}<|im_end|>
<|im_start|>assistant
```

Gemma-2 uses its own turn markers and no system block. When Qwen3's thinking template is on, it additionally opens a `<think>` block at the start of the assistant turn.

The condition content. What follows the assistant header depends on the condition, and it is placed *in the assistant turn*, as if the model had already written it:

- `no_cot`: nothing; the model answers from the question alone.
- `expert_stripped`, `corrupted_expert`, `expert_25/50/75pct`, and the `mid_*` conditions: the worked solution text, verbatim.
- `correct_strategy`, `wrong_strategy`, `adv_*`: a hint lead-in carrying the strategy text, with no worked solution.
- `wrong_strategy_correct`, `wrong_strategy_corrupted`: the hint lead-in and then a worked solution.

The hint lead-in is this exact string, with the strategy substituted for {`hint`}:

```
\n\nHint: {hint}\n\nFollowing the hint, solving step by step:\n
```

So a strategy prompt ends with "`Following the hint, solving step by step:`" and the model continues from there.

B.2 OBJECTS

The example is problem 0 of `synthobjects7`. The same four strategy texts (one correct, three deceptive) are used at every tier of the task; the wrong strategy of a task is the correct strategy of another.

The correct strategy (`correct_strategy`).

```
Here's a strategy for solving this type of problem:  Start
by writing down each person's name and the item they
initially hold, creating a simple list you can reference
and update.  Then, work through each swap one at a time
in the exact order given, updating both people's items
immediately after every single swap before moving on to the
next one.  Don't try to take shortcuts -- process each swap
```

```
sequentially against the current state.  Once you've applied
all the swaps in order, look up the queried person on your
final updated list to find the answer.
```

Deceptive strategy *swaps cancel* (`adv_skip`).

```
Here's a useful shortcut for this type of problem:  In most
cases, the swaps tend to cancel each other out over the
course of the game.  Rather than tracking every single swap,
simply look at what each person started with -- that is
almost always what they end up with.  Save yourself the
effort of tracking and just report the initial assignment
for the queried person.
```

Deceptive strategy *last swap only* (`adv_last_swap`).

```
Here's an efficient strategy for solving this type of
problem:  You don't need to track every swap -- only the
very last swap matters.  If the queried person is involved
in the final swap, they now hold whatever the other person
in that swap originally started with at the beginning of the
game.  If the queried person is NOT involved in the final
swap, they still have whatever they started with.  This
works because earlier swaps get overwritten.
```

Deceptive strategy *parity* (`adv_parity`).

```
Here's a mathematical approach to this type of problem:
Count how many times the queried person appears in the list
of swaps.  If they appear an even number of times (including
zero), they end up with their original item -- the swaps
cancel out in pairs.  If they appear an odd number of times,
they end up with the item of the person they FIRST swapped
with.  This parity rule saves you from tracking the full
chain of swaps.
```

The question, as it appears inside the user message.

```
Alice, Bob, Claire, Dave, Eve, Fred, and Gertrude are
holding a white elephant gift exchange. At the start of
the event, they are each holding a present of a different
color: Alice has a brown present, Bob has a orange present,
Claire has a white present, Dave has a blue present, Eve has
a yellow present, Fred has a green present, and Gertrude has
a red present. As the event progresses, pairs of people swap
gifts. First, Claire and Eve swap their gifts. Then, Alice
and Eve swap their gifts. Then, Bob and Alice swap their
gifts. Then, Alice and Dave swap their gifts. Then, Bob and
Alice swap their gifts. Then, Eve and Dave swap their gifts.
Finally, Alice and Eve swap their gifts. At the end of the
event, Alice has the
Options:
(A) blue present
(B) green present
(C) red present
(D) yellow present
(E) white present
(F) brown present
(G) orange present
```

Correct answer: `(G)`.

The worked solution (`expert_stripped`), placed verbatim after the assistant header.

```
Let me track each swap step by step.

**Initial state:**
- Alice: brown present
- Bob: orange present
- Claire: white present
- Dave: blue present
- Eve: yellow present
- Fred: green present
- Gertrude: red present

**Swap 1: Claire and Eve swap**
- Alice: brown present
- Bob: orange present
- Claire: yellow present
- Dave: blue present
- Eve: white present
- Fred: green present
- Gertrude: red present

**Swap 2: Alice and Eve swap**
- Alice: white present
- Bob: orange present
- Claire: yellow present
- Dave: blue present
- Eve: brown present
- Fred: green present
- Gertrude: red present

**Swap 3: Bob and Alice swap**
- Alice: orange present
- Bob: white present
- Claire: yellow present
- Dave: blue present
- Eve: brown present
- Fred: green present
- Gertrude: red present

**Swap 4: Alice and Dave swap**
- Alice: blue present
- Bob: white present
- Claire: yellow present
- Dave: orange present
- Eve: brown present
- Fred: green present
- Gertrude: red present

**Swap 5: Bob and Alice swap**
- Alice: white present
- Bob: blue present
- Claire: yellow present
- Dave: orange present
- Eve: brown present
- Fred: green present
- Gertrude: red present

**Swap 6: Eve and Dave swap**
- Alice: white present
- Bob: blue present
- Claire: yellow present
- Dave: brown present
- Eve: orange present
- Fred: green present
```

```
- Gertrude: red present

**Swap 7: Alice and Eve swap**
- Alice: orange present
- Bob: blue present
- Claire: yellow present
- Dave: brown present
- Eve: white present
- Fred: green present
- Gertrude: red present
```

The corrupted variant (`corrupted_expert`). Identical except for these lines (2 of 73 differ):

line	expert_stripped	corrupted_expert
67	- Alice: orange present	- Alice: blue present
68	- Bob: blue present	- Bob: orange present

B.3 NAVIGATE

The example is problem 0 of `synthnavigate10`. The same four strategy texts (one correct, three deceptive) are used at every tier of the task; the wrong strategy of a task is the correct strategy of another.

The correct strategy (`correct_strategy`).

```
Here's a strategy for solving this type of navigation
puzzle:  Start by setting up a coordinate system with
your initial position at the origin (0,0) and choose an
initial facing direction.  Track your position step by
step, applying each movement and turn in order, updating
your coordinates and facing direction as you go.  After
completing every instruction, compare your final coordinates
to your starting point (0,0) to determine if you've
returned.
```

Deceptive strategy *forward = backward* (`adv_skip`).

```
Here's a useful shortcut for this type of problem:  In most
navigation sequences, the turns and steps tend to cancel
each other out.  Rather than tracking every single step and
turn, just check if the total number of forward steps equals
the total number of backward steps.  If they match, you
return to the start.  Save yourself the effort of tracking
direction changes.
```

Deceptive strategy *last step only* (`adv_last_step`).

```
Here's an efficient strategy for solving this type of
problem:  You don't need to track every step -- only the
very last movement matters.  If the last instruction is a
step (not a turn), you did NOT return to start.  If the last
instruction is a turn, you DID return to start, because the
steps before it balanced out.
```

Deceptive strategy *count turns* (`adv_count_turns`).

```
Here's a mathematical approach to this type of problem:
Count the total number of turns in the sequence.  If the
number of turns is even, the direction changes cancel out
and you return to the starting point.  If the number of
turns is odd, you end up facing a different direction and
therefore at a different position.  This saves you from
tracking individual steps.
```

The question, as it appears inside the user message.

```
If you follow these instructions, do you return to the
starting point? Take 9 steps. Turn around. Take 9 steps.
Turn right. Turn right. Turn left. Turn left. Turn around.
Turn right. Turn left.
Options:
- Yes
- No
```

Correct answer: `Yes`.

The worked solution (`expert_stripped`), placed verbatim after the assistant header.

```
Starting at position (0, 0), facing North.

Step 1: Take 9 steps -> Position (0, 9)
Step 2: Turn around -> Now facing South
Step 3: Take 9 steps -> Position (0, 0)
Step 4: Turn right -> Now facing West
Step 5: Turn right -> Now facing North
Step 6: Turn left -> Now facing West
Step 7: Turn left -> Now facing South
Step 8: Turn around -> Now facing North
Step 9: Turn right -> Now facing East
Step 10: Turn left -> Now facing North

Final position: (0, 0)
```

The corrupted variant (`corrupted_expert`). Identical except for these lines (2 of 14 differ):

line	expert_stripped	corrupted_expert
5	`Step 3: Take 9 steps -> Position (0, 0)`	`Step 3: Take 9 steps -> Position (3, 0)`
14	`Final position: (0, 0)`	`Final position: (3, 0)`

B.4 DATE

The example is problem 0 of `synthdate_s2`. The same four strategy texts (one correct, three deceptive) are used at every tier of the task; the wrong strategy of a task is the correct strategy of another.

The correct strategy (`correct_strategy`).

```
Here's a strategy for solving this type of date problem:
First, carefully read the scenario to establish what today's
date is -- watch for indirect clues like 'yesterday was
X' (meaning today is X+1) or 'the deadline is in 3 days'
(meaning today is deadline minus 3).  Then, apply the
requested shift (yesterday, one week ago, one year from now,
etc.)  to today's date, being careful with month lengths and
year boundaries.  Finally, format your answer as MM/DD/YYYY.
```

Deceptive strategy `adv_skip`.

```
Here's a useful shortcut for this type of problem:  The
scenario setup is usually just filler -- you can ignore it.
The date mentioned first in the problem is almost always
the answer.  Just take the first date you see, format it as
MM/DD/YYYY, and match it to an option.
```

Deceptive strategy `adv_ignore_shift`.

```
Here's an efficient strategy for this type of problem:  The
shift (yesterday, one week ago, etc.)  is usually negligible
and can be ignored.  Focus on establishing today's date from
the scenario, then just report today's date as the answer.
The shift rarely changes the answer enough to matter.
```

Deceptive strategy `adv_wrong_direction`.

```
Here's a tip for this type of problem:  When the question
says 'ago' or 'before', you need to ADD to the date.  When
it says 'from now' or 'later', you need to SUBTRACT from
the date.  This is because dates count backwards -- earlier
dates have higher numbers.
```

The question, as it appears inside the user message.

```
Yesterday was 5/25/2010. What is the date 24 hours later in MM/DD/YYYY?
Options:
(A) 05/17/2010
(B) 05/27/2009
(C) 05/26/2010
(D) 05/27/2010
(E) 06/03/2010
(F) 06/06/2010
```

Correct answer: `(D)`.

The worked solution (`expert_stripped`), placed verbatim after the assistant header.

```
Step 1: Establish today's date.
Yesterday was May 25, 2010, so today is May 26, 2010.

Step 2: Apply the shift.
The question asks for the date 24 hours later (today + 1 day).
May 26, 2010 + (1 days) = May 27, 2010.

In MM/DD/YYYY format: 05/27/2010
```

The corrupted variant (`corrupted_expert`). Identical except for these lines (2 of 8 differ):

line	expert_stripped	corrupted_expert
6	`May 26, 2010 + (1 days) = May 27, 2010.`	`May 26, 2010 + (1 days) = May 27, 2009.`
8	`In MM/DD/YYYY format:  05/27/2010`	`In MM/DD/YYYY format:  05/27/2009`

C FULL RESULTS UNDER EVERY CONDITION

This appendix prints every cell behind the figures and tables of the body: accuracy under every condition for every model, task and tier (Section C.1, the full version of Table 2); adoption of the planted answer for every condition that plants one (Section C.2); the full outcome breakdown at the hardest tier of each task (Section C.3); and the per-cell version of the strategy-versus-solution comparison of Section 4 (Section C.4). Condition names are those of Table 12 in Appendix A; the `no_cot` column of the accuracy tables is the capability of the model at that tier.

C.1 ACCURACY BY CONDITION AND TIER

Accuracy (%) over all 200 problems of each tier; the count of generations with no extractable answer is in parentheses where it is not zero.

Condition	obj3	obj5	obj7	obj9	obj11
no_cot	82.5	75.5	64.5 (5)	40.5	40.0 (1)
expert_stripped	93.5	91.5	87.5	83.5	76.5
expert_25pct	71.0	84.0	76.0	68.5	62.5
expert_50pct	74.0	89.0	79.0	71.0	65.5
expert_75pct	82.0	89.5	82.5	78.0	67.0
corrupted_expert	5.5	7.0	8.0	8.5	8.5
correct_strategy	78.5	80.5	81.5	50.5 (47)	21.0 (140)
wrong_strategy	77.0	69.5	57.0 (9)	41.0 (19)	29.0 (54)
wrong_strategy_correct	95.0	93.0	91.0	92.0	95.0
wrong_strategy_corrupted	3.0	5.0	3.0	7.0	7.5
reversed_correct	35.0	24.0	17.0 (9)	5.5 (103)	1.0 (160)
reversed_corrupted	17.0	19.0	17.5 (8)	4.0 (104)	1.5 (167)
adv_skip	38.5	14.5	10.5	10.0	6.5
adv_last_swap	64.0	28.0	19.0 (2)	17.0 (1)	12.5
adv_parity	50.5	35.0 (1)	24.0 (2)	19.0 (5)	18.0 (5)

Table 28: Accuracy of Gemma-2-9B on objects.

Condition	obj3	obj5	obj7	obj9	obj11
no_cot	73.0	74.5	66.5	50.5	42.5 (1)
expert_stripped	70.0 (44)	78.5 (22)	85.0 (8)	82.0 (1)	79.5 (5)
expert_25pct	94.0	85.0	88.0	84.0	84.0
expert_50pct	92.5	90.5	91.0	88.0	87.0
expert_75pct	97.0	91.5	91.5	91.0	88.5
corrupted_expert	5.5	1.5 (2)	1.0	1.5	3.0
correct_strategy	90.5	91.5	88.5	75.0	74.5
wrong_strategy	60.5 (3)	48.5 (1)	39.5 (7)	33.5 (6)	27.5 (5)
wrong_strategy_correct	85.0 (1)	76.5 (8)	65.5 (8)	63.5 (2)	60.5 (2)
wrong_strategy_corrupted	2.5 (3)	4.0 (5)	1.5 (1)	5.0 (1)	7.0
reversed_correct	0.0 (200)	0.0 (200)	0.0 (200)	0.0 (200)	0.0 (195)
reversed_corrupted	9.5 (176)	0.0 (200)	0.0 (199)	0.0 (199)	0.0 (190)
adv_skip	49.0 (1)	36.5	24.5 (3)	23.0 (4)	21.5 (5)
adv_last_swap	55.0	41.5	25.0 (2)	30.0 (4)	17.0 (8)
adv_parity	28.0 (32)	26.0 (19)	20.5 (12)	11.0 (23)	15.0 (17)

Table 29: Accuracy of Llama-3.1-8B on objects.

C.2 ADOPTION OF THE PLANTED ANSWER

Adoption (%) is the share of answers equal to the specific answer the planted content implies, over the problems where that answer differs from the correct one (n in parentheses where it is below 200; freebies removed). `mid_corrupt_stop` and `mid_corrupt_full` were run on Qwen3-14B only.

Condition	obj3	obj5	obj7	obj9	obj11
no_cot	82.5	82.0	68.0	72.0	61.5
expert_stripped	100.0	99.0	99.5	98.0	96.5
expert_25pct	86.0	81.5	86.0	77.5	69.0
expert_50pct	89.0	84.5	88.5	83.0	77.5
expert_75pct	95.5	91.0	96.5	89.0	81.5
corrupted_expert	0.0	3.0	2.0	5.5	8.0
correct_strategy	89.0	87.0	85.0	84.0	67.0
wrong_strategy	71.0 (15)	64.5 (11)	58.5 (5)	50.0 (17)	46.5 (19)
wrong_strategy_correct	100.0	100.0	98.5	96.5	94.0
wrong_strategy_corrupted	0.5	3.5	3.0	7.0	9.5
reversed_correct	90.5	93.0	96.0	87.5	80.5
reversed_corrupted	80.0	83.0	92.0	81.5	78.5
adv_skip	42.0	52.0	38.0	27.5	15.0
adv_last_swap	24.5	23.5	20.5	19.5	13.0
adv_parity	23.5	27.0	17.5 (1)	21.0	18.0

Table 30: Accuracy of Mistral-Small-22B on objects.

Condition	obj3	obj5	obj7	obj9	obj11
no_cot	87.0	88.0	90.5	84.5	78.0
expert_stripped	91.0	92.5	98.0	96.0	92.0
expert_25pct	91.5	92.0	97.0	93.0	89.5
expert_50pct	86.5	91.0	96.5	94.0	88.5
expert_75pct	90.5	92.0	96.5	94.0	90.0
corrupted_expert	6.0	4.5	1.5	2.0	3.0
correct_strategy	87.0	91.5	89.5	87.0	83.5
wrong_strategy	87.5	90.5	88.5	85.5	81.5
wrong_strategy_correct	91.0	89.5	96.0	92.0	89.0
wrong_strategy_corrupted	10.0	4.5	1.5	3.5	3.5
reversed_correct	89.0	90.0	89.0	55.0	31.0
reversed_corrupted	78.0	77.0	76.5	35.5	26.5
adv_skip	76.5	54.0	37.5	31.0	27.5
adv_last_swap	87.0	85.0	77.0	49.0	44.5
adv_parity	77.5	83.0	65.0	48.5	32.0

Table 31: Accuracy of Qwen-2.5-14B on objects.

Condition	obj3	obj5	obj7	obj9	obj11
no_cot	100.0	100.0	99.5	95.5	93.0
expert_stripped	100.0	100.0	100.0	99.5	100.0
expert_25pct	100.0	99.5	100.0	98.5	97.0
expert_50pct	100.0	100.0	100.0	99.5	97.5
expert_75pct	100.0	100.0	100.0	100.0	98.0
corrupted_expert	24.0	5.5	1.5	0.0	1.0
correct_strategy	100.0	99.5	97.5	96.5	91.0
wrong_strategy	11.5 (35)	11.0 (22)	8.0 (38)	4.5 (29)	3.0 (42)
wrong_strategy_correct	100.0	99.5	100.0	99.5	99.5
wrong_strategy_corrupted	18.0	4.0	0.0	0.0	0.5
reversed_correct	99.0	99.5	99.0	97.5	94.5
reversed_corrupted	99.5	99.0	99.5	98.5	94.5
adv_skip	70.5	49.0	30.5	26.5	23.5
adv_last_swap	82.5	66.5	64.0	61.5	57.0
adv_parity	59.0	50.0	38.5	29.0	28.0

Table 32: Accuracy of Qwen3-14B, thinking off on objects.

Condition	nav5	nav10	nav15	nav20
no_cot	81.5	56.5 (2)	50.5	43.0 (8)
expert_stripped	100.0	99.5	100.0	100.0
expert_25pct	90.0	79.0	55.0	77.0
expert_50pct	93.5	82.0	57.0	80.5
expert_75pct	95.0	88.0	69.5	84.0
corrupted_expert	0.0	0.0	0.5	29.5
correct_strategy	83.5 (6)	82.5 (5)	59.5 (2)	76.5 (8)
wrong_strategy	78.5 (1)	47.0 (35)	44.0 (32)	45.5 (48)
wrong_strategy_correct	82.0 (36)	46.0 (108)	59.0 (82)	56.5 (87)
wrong_strategy_corrupted	0.0 (43)	0.5 (73)	0.5 (61)	0.5 (29)
reversed_correct	76.5 (26)	90.0 (10)	95.5 (3)	96.0
reversed_corrupted	18.0 (61)	16.0 (35)	13.5 (41)	15.5 (40)
adv_skip	80.0 (4)	77.0 (6)	46.5 (16)	57.0 (31)
adv_last_step	59.5	64.0	46.5 (4)	52.5 (13)
adv_count_turns	35.5 (13)	48.0 (24)	49.5 (14)	51.0 (7)

Table 33: Accuracy of Gemma-2-9B on navigate.

Condition	nav5	nav10	nav15	nav20
no_cot	74.0	57.0	68.0	58.5
expert_stripped	92.0	88.5	100.0	99.0
expert_25pct	77.0	71.5	66.5	65.5
expert_50pct	85.5	73.0	72.5	64.5
expert_75pct	85.0	79.0	85.5	74.5
corrupted_expert	5.0	0.0	0.0	0.0
correct_strategy	90.0	81.0	66.0	76.5
wrong_strategy	23.0 (41)	10.0 (91)	15.5 (78)	23.0 (34)
wrong_strategy_correct	92.0	88.5	100.0	96.0
wrong_strategy_corrupted	5.5	0.0	0.0	0.0
reversed_correct	83.5	89.0	98.0	79.5
reversed_corrupted	11.5	2.0	2.0	0.0
adv_skip	79.0	85.0	63.0	76.0
adv_last_step	42.5	68.5	47.5	70.5
adv_count_turns	40.5	55.0	42.5	44.0

Table 34: Accuracy of Mistral-Small-22B on navigate.

Condition	nav5	nav10	nav15	nav20
no_cot	71.0 (1)	69.5	52.5	67.5 (2)
expert_stripped	48.5 (91)	48.5 (98)	50.5 (96)	49.0 (84)
expert_25pct	60.5 (3)	68.5 (33)	52.0 (4)	65.0 (12)
expert_50pct	75.5 (4)	64.5 (40)	51.0 (1)	64.0 (17)
expert_75pct	69.5 (35)	68.5 (41)	58.5 (2)	68.0 (17)
corrupted_expert	3.0 (91)	0.5 (100)	1.5 (123)	15.5 (109)
correct_strategy	73.5 (6)	77.5 (4)	55.0 (1)	69.5 (9)
wrong_strategy	63.0 (16)	58.0 (25)	60.0	51.0 (11)
wrong_strategy_correct	0.0 (200)	0.0 (200)	0.0 (200)	0.0 (200)
wrong_strategy_corrupted	0.0 (200)	0.0 (200)	0.0 (200)	0.0 (200)
reversed_correct	28.5 (127)	12.5 (173)	15.5 (166)	14.5 (167)
reversed_corrupted	6.5 (149)	2.0 (193)	3.0 (187)	5.5 (188)
adv_skip	80.5 (10)	73.5 (4)	56.5 (2)	69.5 (5)
adv_last_step	50.5 (2)	68.0 (1)	42.0 (2)	61.5 (11)
adv_count_turns	41.0 (19)	53.5 (29)	28.5 (13)	51.0 (7)

Table 35: Accuracy of Llama-3.1-8B on navigate.

Condition	nav5	nav10	nav15	nav20
no_cot	87.0	85.5	74.5	77.5 (1)
expert_stripped	96.0	97.0	100.0	98.0
expert_25pct	95.0	86.5	76.5	85.0
expert_50pct	93.5	87.5	77.5	84.0
expert_75pct	95.5	88.5	89.0	90.0
corrupted_expert	15.0	1.5	2.0	2.5
correct_strategy	96.0	90.5	70.5	81.5 (1)
wrong_strategy	89.5	83.0	70.0	66.5 (2)
wrong_strategy_correct	93.0	95.0	97.5	88.0
wrong_strategy_corrupted	19.0	4.5	2.0	3.5
reversed_correct	92.5	80.0	98.0	87.0
reversed_corrupted	30.5	6.0	4.0	1.0
adv_skip	94.0	93.5	65.5	83.5
adv_last_step	86.5	77.5	60.0	77.0
adv_count_turns	84.5	79.5	66.5	69.5

Table 36: Accuracy of Qwen-2.5-14B on navigate.

Condition	nav5	nav10	nav15	nav20
no_cot	90.0	98.5	98.5	94.5
expert_stripped	100.0	100.0	100.0	100.0
expert_25pct	93.5	98.5	87.5	94.5
expert_50pct	98.0	99.0	92.0	95.0
expert_75pct	97.0	98.5	92.0	96.5
corrupted_expert	1.0	0.5	0.0	0.0 (1)
correct_strategy	99.5	97.5	89.5	92.5
wrong_strategy	5.0 (168)	0.5 (188)	3.5 (180)	6.0 (168)
wrong_strategy_correct	100.0	100.0	100.0	100.0
wrong_strategy_corrupted	1.5	0.0	0.0	0.0
reversed_correct	100.0	100.0	100.0	100.0
reversed_corrupted	4.0	0.0	0.0	0.0
adv_skip	88.0	92.5	62.0	81.0 (1)
adv_last_step	67.0	69.0	47.0	70.0
adv_count_turns	32.0	70.0	27.5	62.5

Table 37: Accuracy of Qwen3-14B, thinking off on navigate.

Condition	s1	s2	s3
no_cot	91.0 (4)	56.0 (23)	18.5 (87)
expert_stripped	61.5 (75)	35.5 (126)	11.0 (177)
expert_25pct	90.0 (5)	60.5 (20)	33.5 (58)
expert_50pct	88.5 (16)	87.0 (5)	51.0 (41)
expert_75pct	93.5 (6)	85.5 (9)	93.0 (3)
corrupted_expert	0.5 (152)	0.0 (140)	0.0 (181)
correct_strategy	94.0 (6)	62.5 (19)	20.5 (86)
wrong_strategy	89.5 (7)	52.0 (37)	16.5 (95)
wrong_strategy_correct	26.0 (148)	19.5 (159)	30.5 (138)
wrong_strategy_corrupted	0.0 (194)	0.5 (182)	0.0 (176)
reversed_correct	0.0 (200)	0.0 (200)	50.0 (90)
reversed_corrupted	0.0 (200)	0.0 (200)	50.0 (90)
adv_skip	77.5 (5)	48.5 (24)	10.0 (100)
adv_ignore_shift	80.0 (18)	55.5 (23)	22.5 (77)
adv_wrong_direction	83.5 (13)	58.0 (25)	18.0 (89)

Table 38: Accuracy of Llama-3.1-8B on date.

Condition	s1	s2	s3
no_cot	98.5	75.5 (12)	36.0 (59)
expert_stripped	100.0	99.0	99.0
expert_25pct	98.0	80.0 (16)	56.5 (35)
expert_50pct	95.5 (1)	90.0	58.0 (38)
expert_75pct	97.5 (1)	93.5	92.5 (1)
corrupted_expert	2.0 (1)	2.0 (1)	0.5
correct_strategy	98.5	80.5 (17)	38.5 (86)
wrong_strategy	99.5	74.0 (13)	33.0 (74)
wrong_strategy_correct	99.5	98.0	99.5
wrong_strategy_corrupted	0.5 (2)	1.5 (4)	0.5 (1)
reversed_correct	1.0 (198)	16.5 (164)	0.0 (200)
reversed_corrupted	4.5 (186)	7.0 (153)	0.0 (200)
adv_skip	55.0 (32)	40.0 (29)	3.0 (153)
adv_ignore_shift	96.0 (1)	64.5 (22)	35.0 (74)
adv_wrong_direction	94.5 (6)	66.5 (25)	27.0 (92)

Table 39: Accuracy of Gemma-2-9B on date.

Condition	s1	s2	s3
no_cot	94.0	70.5 (2)	37.5 (42)
expert_stripped	99.0	97.0	89.0 (17)
expert_25pct	95.0 (2)	68.0 (16)	28.5 (62)
expert_50pct	75.0 (10)	95.0	41.0 (41)
expert_75pct	98.0	93.5	91.0 (1)
corrupted_expert	0.0	2.5	0.5 (20)
correct_strategy	98.0	83.5 (5)	34.5 (71)
wrong_strategy	11.0 (93)	7.5 (78)	22.0 (93)
wrong_strategy_correct	99.0	97.0	79.0 (27)
wrong_strategy_corrupted	0.0	1.0	1.0 (47)
reversed_correct	100.0	90.5 (13)	96.0
reversed_corrupted	36.5 (3)	4.0 (30)	28.5
adv_skip	84.0 (1)	56.0 (6)	23.0 (50)
adv_ignore_shift	91.5 (7)	71.5 (11)	35.0 (79)
adv_wrong_direction	90.5 (7)	62.0 (20)	36.0 (50)

Table 40: Accuracy of Mistral-Small-22B on date.

Condition	s1	s2	s3
no_cot	98.5	81.0	67.5
expert_stripped	99.5	96.5	97.5
expert_25pct	96.5	92.5	63.0
expert_50pct	98.5	92.5	71.5 (2)
expert_75pct	97.5	92.0	95.0
corrupted_expert	64.0	38.5	28.0
correct_strategy	98.5	89.0	53.5 (1)
wrong_strategy	98.5	84.5	60.5 (4)
wrong_strategy_correct	99.0	97.0	97.5
wrong_strategy_corrupted	69.0	35.0	25.5
reversed_correct	97.5	88.0	91.5
reversed_corrupted	81.0	49.0	64.0
adv_skip	98.0	85.0	55.0 (3)
adv_ignore_shift	96.5	91.5	59.5 (2)
adv_wrong_direction	97.5	80.5 (1)	58.5 (2)

Table 41: Accuracy of Qwen-2.5-14B on date.

Condition	s1	s2	s3
no_cot	98.0	91.5 (1)	73.0 (9)
expert_stripped	100.0	100.0	100.0
expert_25pct	99.5	96.5	77.5 (1)
expert_50pct	95.5 (1)	97.0	87.5
expert_75pct	100.0	97.5	99.0
corrupted_expert	1.5	0.0	1.5
correct_strategy	100.0	91.5 (1)	71.0 (4)
wrong_strategy	10.5 (69)	10.0 (57)	12.0 (51)
wrong_strategy_correct	100.0	99.5	100.0
wrong_strategy_corrupted	1.0	0.0	0.0
reversed_correct	66.5 (67)	84.5 (31)	98.5
reversed_corrupted	37.5 (71)	19.5 (40)	14.0
adv_skip	95.0	80.0	64.5 (1)
adv_ignore_shift	99.0	97.0	71.0 (4)
adv_wrong_direction	96.5 (2)	86.5 (1)	67.0 (2)

Table 42: Accuracy of Qwen3-14B, thinking off on date.

Condition	obj3	obj5	obj7	obj9	obj11
no_cot	100.0	100.0	99.5	100.0	99.0
expert_stripped	100.0	100.0	100.0	100.0	99.5
expert_25pct	100.0	100.0	99.5	99.5	98.0
expert_50pct	100.0	100.0	100.0	98.5	98.5
expert_75pct	100.0	100.0	100.0	100.0	98.5
corrupted_expert	99.0	92.5	79.5	60.5	61.5
correct_strategy	100.0	100.0	99.5	99.5	95.5
wrong_strategy	98.0 (4)	97.0 (6)	98.0 (2)	95.0 (4)	94.0 (5)
wrong_strategy_correct	100.0	100.0	98.5	100.0	99.0
wrong_strategy_corrupted	99.0	97.5	77.5	67.0	57.0
reversed_correct	100.0	100.0	99.5	99.5	98.5
reversed_corrupted	100.0	100.0	99.5	97.5	98.0
adv_skip	100.0	98.5	97.0	92.5	85.5
adv_last_swap	97.0 (1)	99.0	90.5	83.5	78.5 (1)
adv_parity	96.5 (1)	97.5	94.0	81.0	74.5

Table 43: Accuracy of Qwen3-14B with thinking on, objects.

Condition	nav5	nav10	nav15	nav20
no_cot	96.0	100.0	99.5	98.0
expert_stripped	100.0	100.0	99.5	100.0
expert_25pct	97.0	97.0	86.5	92.5
expert_50pct	98.0	98.0	91.0	95.0
expert_75pct	96.0	100.0	94.5	96.5
corrupted_expert	90.0	82.0 (1)	81.5	68.0
correct_strategy	100.0	97.5 (1)	96.5	98.0
wrong_strategy	81.0 (6)	91.5 (6)	87.5 (7)	89.5 (2)
wrong_strategy_correct	100.0	99.5	99.5	100.0
wrong_strategy_corrupted	92.5	94.0	86.0 (1)	84.5 (1)
reversed_correct	99.0	99.5	100.0	99.5
reversed_corrupted	88.5	91.5	91.0	84.0
adv_skip	98.0	98.5 (1)	86.5 (2)	91.0 (6)
adv_last_step	92.0 (1)	95.5 (1)	84.0	93.0 (2)
adv_count_turns	82.5 (1)	95.0	78.0	89.5 (2)

Table 44: Accuracy of Qwen3-14B with thinking on, navigate.

Condition	s1	s2	s3
no_cot	100.0	97.0	97.0
expert_stripped	100.0	99.5	98.5
expert_25pct	100.0	93.5 (1)	93.5 (6)
expert_50pct	98.5	96.5	91.0 (2)
expert_75pct	100.0	97.5	98.5
corrupted_expert	89.5	87.0	77.5
correct_strategy	100.0	100.0	96.0
wrong_strategy	99.5	94.0	97.0 (1)
wrong_strategy_correct	100.0	99.0	98.5 (1)
wrong_strategy_corrupted	95.5	93.0	84.0
reversed_correct	99.5	98.5 (2)	98.0
reversed_corrupted	97.0	90.5 (1)	86.0 (1)
adv_skip	98.5	88.5 (1)	86.5 (5)
adv_ignore_shift	100.0	99.5	95.5
adv_wrong_direction	97.5 (1)	87.5 (1)	81.5 (13)

Table 45: Accuracy of Qwen3-14B with thinking on, date.

condition	obj3	obj5	obj7	obj9	obj11
corrupted_expert	91.5	91.0	76.0	68.5	61.0
wrong_strategy_corrupted	95.0	92.0	88.0	77.5	78.0
adv_skip	85.6 (132)	100.0 (171)	100.0 (179)	100.0 (180)	99.5 (187)
adv_last_swap	19.0	73.8 (168)	75.6 (168)	79.1 (172)	80.2 (182)
adv_parity	26.8 (168)	49.6 (141)	54.1 (157)	58.5 (164)	57.1 (156)

Table 46: Adoption on objects: Gemma-2-9B.

condition	obj3	obj5	obj7	obj9	obj11
corrupted_expert	93.5	95.0	97.5	95.5	92.5
wrong_strategy_corrupted	91.0	87.0	90.5	85.0	75.0
adv_skip	17.4 (132)	35.7 (171)	32.4 (179)	45.0 (180)	43.9 (187)
adv_last_swap	31.5	35.7 (168)	40.5 (168)	33.7 (172)	36.3 (182)
adv_parity	26.2 (168)	34.8 (141)	29.3 (157)	34.1 (164)	35.9 (156)

Table 47: Adoption on objects: Llama-3.1-8B.

condition	obj3	obj5	obj7	obj9	obj11
corrupted_expert	100.0	97.0	98.0	94.5	91.0
wrong_strategy_corrupted	99.5	96.5	96.5	91.5	89.0
adv_skip	81.1 (132)	36.3 (171)	40.8 (179)	66.7 (180)	76.5 (187)
adv_last_swap	70.0	70.8 (168)	63.7 (168)	69.2 (172)	72.0 (182)
adv_parity	54.2 (168)	34.8 (141)	40.8 (157)	51.2 (164)	50.0 (156)

Table 48: Adoption on objects: Mistral-Small-22B.

condition	obj3	obj5	obj7	obj9	obj11
corrupted_expert	89.0	93.5	97.0	93.5	89.0
wrong_strategy_corrupted	86.5	92.5	96.0	90.5	85.5
adv_skip	15.2 (132)	28.7 (171)	21.2 (179)	33.3 (180)	40.6 (187)
adv_last_swap	5.5	8.3 (168)	10.7 (168)	19.2 (172)	22.0 (182)
adv_parity	15.5 (168)	12.8 (141)	22.9 (157)	29.9 (164)	46.2 (156)

Table 49: Adoption on objects: Qwen-2.5-14B.

condition	obj3	obj5	obj7	obj9	obj11
corrupted_expert	76.0	94.5	98.5	99.5	98.5
wrong_strategy_corrupted	82.0	96.0	100.0	99.0	97.5
mid_corrupt_stop	47.5	82.0	91.0	91.5	90.5
mid_corrupt_full	0.0	0.0	0.0	0.0	0.5
adv_skip	40.2 (132)	54.4 (171)	68.7 (179)	64.4 (180)	61.5 (187)
adv_last_swap	17.0	36.9 (168)	33.9 (168)	26.7 (172)	26.4 (182)
adv_parity	38.7 (168)	59.6 (141)	60.5 (157)	72.0 (164)	72.4 (156)

Table 50: Adoption on objects: Qwen3-14B OFF.

condition	obj3	obj5	obj7	obj9	obj11
corrupted_expert	1.0	7.5	20.5	38.5	38.5
wrong_strategy_corrupted	1.0	2.5	21.5	32.0	41.0
mid_corrupt_stop	0.5	2.0	22.0	32.5	40.5
mid_corrupt_full	0.5	1.5	0.5	0.0	0.0
adv_skip	0.0 (132)	0.0 (171)	0.6 (179)	1.7 (180)	2.7 (187)
adv_last_swap	2.0	0.6 (168)	5.4 (168)	7.0 (172)	6.6 (182)
adv_parity	3.6 (168)	3.5 (141)	5.7 (157)	19.5 (164)	24.4 (156)

Table 51: Adoption on objects: Qwen3-14B ON.

condition	nav5	nav10	nav15	nav20
corrupted_expert	100.0	100.0	99.5	70.5
wrong_strategy_corrupted	78.5†	63.0†	69.0†	85.0
adv_skip	28.6 (84)	35.4 (99)	91.0 (100)	54.0 (100)
adv_last_step	74.8 (103)	93.5 (62)	81.1 (106)	78.3 (60)
adv_count_turns	69.3 (153)	56.2 (48)	45.8 (155)	51.9 (52)

Table 52: Adoption on navigate: Gemma-2-9B.

condition	nav5	nav10	nav15	nav20
corrupted_expert	51.5†	49.5†	37.0†	30.0†
wrong_strategy_corrupted	0.0†	0.0†	0.0†	0.0†
adv_skip	23.8 (84)	37.4 (99)	75.0 (100)	44.0 (100)
adv_last_step	70.9 (103)	85.5 (62)	95.3 (106)	95.0 (60)
adv_count_turns	58.2 (153)	68.8 (48)	79.4 (155)	82.7 (52)

Table 53: Adoption on navigate: Llama-3.1-8B.

condition	nav5	nav10	nav15	nav20
corrupted_expert	95.0	100.0	100.0	100.0
wrong_strategy_corrupted	94.5	100.0	100.0	100.0
adv_skip	36.9 (84)	29.3 (99)	73.0 (100)	46.0 (100)
adv_last_step	98.1 (103)	95.2 (62)	90.6 (106)	93.3 (60)
adv_count_turns	75.2 (153)	60.4 (48)	61.9 (155)	57.7 (52)

Table 54: Adoption on navigate: Mistral-Small-22B.

condition	nav5	nav10	nav15	nav20
corrupted_expert	85.0	98.5	98.0	97.5
wrong_strategy_corrupted	81.0	95.5	98.0	96.5
adv_skip	3.6 (84)	11.1 (99)	68.0 (100)	33.0 (100)
adv_last_step	15.5 (103)	58.1 (62)	65.1 (106)	63.3 (60)
adv_count_turns	20.3 (153)	2.1 (48)	42.6 (155)	1.9 (52)

Table 55: Adoption on navigate: Qwen-2.5-14B.

condition	nav5	nav10	nav15	nav20
corrupted_expert	99.0	99.5	100.0	99.5
wrong_strategy_corrupted	98.5	100.0	100.0	100.0
mid_corrupt_stop	54.1 (85)	51.5 (99)	78.2 (101)	60.4 (101)
mid_corrupt_full	28.2 (85)	3.0 (99)	1.0 (101)	1.0 (101)
adv_skip	19.0 (84)	12.1 (99)	76.0 (100)	36.0 (100)
adv_last_step	64.1 (103)	100.0 (62)	99.1 (106)	100.0 (60)
adv_count_turns	88.2 (153)	100.0 (48)	93.5 (155)	94.2 (52)

Table 56: Adoption on navigate: Qwen3-14B OFF.

condition	nav5	nav10	nav15	nav20
corrupted_expert	10.0	17.5	18.5	32.0
wrong_strategy_corrupted	7.5	6.0	13.5	15.0
mid_corrupt_stop	5.9 (85)	5.1 (99)	7.9 (101)	13.9 (101)
mid_corrupt_full	0.0 (85)	3.0 (99)	2.0 (101)	2.0 (101)
adv_skip	1.2 (84)	2.0 (99)	25.0 (100)	12.0 (100)
adv_last_step	12.6 (103)	11.3 (62)	21.7 (106)	16.7 (60)
adv_count_turns	22.2 (153)	10.4 (48)	28.4 (155)	5.8 (52)

Table 57: Adoption on navigate: Qwen3-14B ON.

condition	s1	s2	s3
corrupted_expert	97.0	97.0	99.5
wrong_strategy_corrupted	98.0	96.0	99.0

Table 58: Adoption on date: Gemma-2-9B.

condition	s1	s2	s3
corrupted_expert	23.0†	30.0†	9.5†
wrong_strategy_corrupted	3.0†	8.5†	12.0†

Table 59: Adoption on date: Llama-3.1-8B.

condition	s1	s2	s3
corrupted_expert	99.5	93.5	87.5
wrong_strategy_corrupted	99.0	97.0	74.0†

Table 60: Adoption on date: Mistral-Small-22B.

condition	s1	s2	s3
corrupted_expert	33.5	55.5	71.0
wrong_strategy_corrupted	30.0	58.0	72.5

Table 61: Adoption on date: Qwen-2.5-14B.

condition	s1	s2	s3
corrupted_expert	98.5	100.0	98.5
wrong_strategy_corrupted	99.0	100.0	100.0
mid_corrupt_stop	21.0	33.5	96.5
mid_corrupt_full	0.0	0.0	0.0

Table 62: Adoption on date: Qwen3-14B OFF.

condition	s1	s2	s3
corrupted_expert	10.0	11.5	19.5
wrong_strategy_corrupted	4.5	5.5	13.0
mid_corrupt_stop	1.5	5.5	34.5
mid_corrupt_full	0.0	6.0	33.0

Table 63: Adoption on date: Qwen3-14B ON.

C.3 Outcome breakdown at the hardest tier

For every wrong-content condition at the hardest tier: adoption, correct, other (with the chance share of the other bin in parentheses) and no-answer, all as % of n. Table 67 collects the cells the body quotes most often in one place.

Table 64: Outcome breakdown at the hardest tier, objects.

model	condition	n_used	adoption	gold	other (chance)	na
Gemma-2-9B	corrupted_expert	200	61.0	8.5	30.5 (11.1)	0.0
Gemma-2-9B	wrong_strategy_corrupted	200	78.0	7.5	14.5 (11.1)	0.0
Gemma-2-9B	adv_skip	187	99.5	0.0	0.5 (11.1)	0.0
Gemma-2-9B	adv_last_swap	182	80.2	4.4	15.4 (11.1)	0.0
Gemma-2-9B	adv_parity	156	57.1	5.8	34.6 (11.1)	2.6
Llama-3.1-8B	corrupted_expert	200	92.5	3.0	4.5 (11.1)	0.0
Llama-3.1-8B	wrong_strategy_corrupted	200	75.0	7.0	18.0 (11.1)	0.0
Llama-3.1-8B	adv_skip	187	43.9	20.9	32.6 (11.1)	2.7
Llama-3.1-8B	adv_last_swap	182	36.3	15.4	45.1 (11.1)	3.3
Llama-3.1-8B	adv_parity	156	35.9	7.7	47.4 (11.1)	9.0
Mistral-Small-22B	corrupted_expert	200	91.0	8.0	1.0 (11.1)	0.0
Mistral-Small-22B	wrong_strategy_corrupted	200	89.0	9.5	1.5 (11.1)	0.0
Mistral-Small-22B	adv_skip	187	76.5	9.1	14.4 (11.1)	0.0
Mistral-Small-22B	adv_last_swap	182	72.0	7.7	20.3 (11.1)	0.0
Mistral-Small-22B	adv_parity	156	50.0	4.5	45.5 (11.1)	0.0
Qwen-2.5-14B	corrupted_expert	200	89.0	3.0	8.0 (11.1)	0.0
Qwen-2.5-14B	wrong_strategy_corrupted	200	85.5	3.5	11.0 (11.1)	0.0
Qwen-2.5-14B	adv_skip	187	40.6	24.6	34.8 (11.1)	0.0
Qwen-2.5-14B	adv_last_swap	182	22.0	44.5	33.5 (11.1)	0.0
Qwen-2.5-14B	adv_parity	156	46.2	16.7	37.2 (11.1)	0.0
Qwen3-14B OFF	corrupted_expert	200	98.5	1.0	0.5 (11.1)	0.0
Qwen3-14B OFF	wrong_strategy_corrupted	200	97.5	0.5	2.0 (11.1)	0.0
Qwen3-14B OFF	mid_corrupt_stop	200	90.5	7.5	2.0 (11.1)	0.0
Qwen3-14B OFF	mid_corrupt_full	200	0.5	99.0	0.5 (11.1)	0.0
Qwen3-14B OFF	adv_skip	187	61.5	20.3	18.2 (11.1)	0.0
Qwen3-14B OFF	adv_last_swap	182	26.4	54.9	18.7 (11.1)	0.0
Qwen3-14B OFF	adv_parity	156	72.4	12.8	14.7 (11.1)	0.0
Qwen3-14B ON	corrupted_expert	200	38.5	61.5	0.0 (11.1)	0.0
Qwen3-14B ON	wrong_strategy_corrupted	200	41.0	57.0	2.0 (11.1)	0.0
Qwen3-14B ON	mid_corrupt_stop	200	40.5	57.0	2.5 (11.1)	0.0
Qwen3-14B ON	mid_corrupt_full	200	0.0	99.5	0.5 (11.1)	0.0
Qwen3-14B ON	adv_skip	187	2.7	85.6	11.8 (11.1)	0.0
Qwen3-14B ON	adv_last_swap	182	6.6	78.6	14.8 (11.1)	0.0
Qwen3-14B ON	adv_parity	156	24.4	69.9	5.8 (11.1)	0.0

C.4 What a strategy is worth, per model and tier

Table 68 contains all intervals and CIs of table 4

Section 4 says that a correct strategy captures about 40% of the gain of the full worked solution. Figure 2 is the per-cell version of that figure: for every model and tier with at least 10 points of headroom, the gain of `correct_strategy` over `no_cot` divided by the gain of `expert_stripped` over `no_cot`. The share varies a good deal across cells, and the trend with capability is carried by objects alone. Over the 20 cells with at least 10 points of headroom and no no-answer generation the median share is 0.41 (IQR 0.20–0.60, range −0.36 to 1.00); on all 38 cells with that headroom it is 0.38. By task the medians are 0.31 on Objects, 0.60 on Navigate and 0.52 on the single Date cell, so the pooled figure is carried by Navigate

model	condition	n_used	adoption	gold	other (chance)	na
Gemma-2-9B	corrupted_expert	200	70.5	29.5	0.0 (0.0)	0.0
Gemma-2-9B	wrong_strategy_corrupted	200	85.0	0.5	0.0 (0.0)	14.5
Gemma-2-9B	adv_skip	100	54.0	36.0	0.0 (0.0)	10.0
Gemma-2-9B	adv_last_step	60	78.3	16.7	0.0 (0.0)	5.0
Gemma-2-9B	adv_count_turns	52	51.9	48.1	0.0 (0.0)	0.0
Llama-3.1-8B	corrupted_expert	200	30.0	15.5	0.0 (0.0)	54.5
Llama-3.1-8B	wrong_strategy_corrupted	200	0.0	0.0	0.0 (0.0)	100.0
Llama-3.1-8B	adv_skip	100	44.0	53.0	0.0 (0.0)	3.0
Llama-3.1-8B	adv_last_step	60	95.0	5.0	0.0 (0.0)	0.0
Llama-3.1-8B	adv_count_turns	52	82.7	17.3	0.0 (0.0)	0.0
Mistral-Small-22B	corrupted_expert	200	100.0	0.0	0.0 (0.0)	0.0
Mistral-Small-22B	wrong_strategy_corrupted	200	100.0	0.0	0.0 (0.0)	0.0
Mistral-Small-22B	adv_skip	100	46.0	54.0	0.0 (0.0)	0.0
Mistral-Small-22B	adv_last_step	60	93.3	6.7	0.0 (0.0)	0.0
Mistral-Small-22B	adv_count_turns	52	57.7	42.3	0.0 (0.0)	0.0
Qwen-2.5-14B	corrupted_expert	200	97.5	2.5	0.0 (0.0)	0.0
Qwen-2.5-14B	wrong_strategy_corrupted	200	96.5	3.5	0.0 (0.0)	0.0
Qwen-2.5-14B	adv_skip	100	33.0	67.0	0.0 (0.0)	0.0
Qwen-2.5-14B	adv_last_step	60	63.3	36.7	0.0 (0.0)	0.0
Qwen-2.5-14B	adv_count_turns	52	1.9	98.1	0.0 (0.0)	0.0
Qwen3-14B OFF	corrupted_expert	200	99.5	0.0	0.0 (0.0)	0.5
Qwen3-14B OFF	wrong_strategy_corrupted	200	100.0	0.0	0.0 (0.0)	0.0
Qwen3-14B OFF	mid_corrupt_stop	101	60.4	39.6	0.0 (0.0)	0.0
Qwen3-14B OFF	mid_corrupt_full	101	1.0	99.0	0.0 (0.0)	0.0
Qwen3-14B OFF	adv_skip	100	36.0	63.0	0.0 (0.0)	1.0
Qwen3-14B OFF	adv_last_step	60	100.0	0.0	0.0 (0.0)	0.0
Qwen3-14B OFF	adv_count_turns	52	94.2	5.8	0.0 (0.0)	0.0
Qwen3-14B ON	corrupted_expert	200	32.0	68.0	0.0 (0.0)	0.0
Qwen3-14B ON	wrong_strategy_corrupted	200	15.0	84.5	0.0 (0.0)	0.5
Qwen3-14B ON	mid_corrupt_stop	101	13.9	86.1	0.0 (0.0)	0.0
Qwen3-14B ON	mid_corrupt_full	101	2.0	98.0	0.0 (0.0)	0.0
Qwen3-14B ON	adv_skip	100	12.0	87.0	0.0 (0.0)	1.0
Qwen3-14B ON	adv_last_step	60	16.7	83.3	0.0 (0.0)	0.0
Qwen3-14B ON	adv_count_turns	52	5.8	94.2	0.0 (0.0)	0.0

Table 65: Outcome breakdown at the hardest tier, navigate.

model	condition	n_used	adoption	gold	other (chance)	na
Gemma-2-9B	corrupted_expert	200	99.5	0.5	0.0 (25.0)	0.0
Gemma-2-9B	wrong_strategy_corrupted	200	99.0	0.5	0.0 (25.0)	0.5
Llama-3.1-8B	corrupted_expert	200	9.5	0.0	0.0 (25.0)	90.5
Llama-3.1-8B	wrong_strategy_corrupted	200	12.0	0.0	0.0 (25.0)	88.0
Mistral-Small-22B	corrupted_expert	200	87.5	0.5	2.0 (25.0)	10.0
Mistral-Small-22B	wrong_strategy_corrupted	200	74.0	1.0	1.5 (25.0)	23.5
Qwen-2.5-14B	corrupted_expert	200	71.0	28.0	1.0 (25.0)	0.0
Qwen-2.5-14B	wrong_strategy_corrupted	200	72.5	25.5	2.0 (25.0)	0.0
Qwen3-14B OFF	corrupted_expert	200	98.5	1.5	0.0 (25.0)	0.0
Qwen3-14B OFF	wrong_strategy_corrupted	200	100.0	0.0	0.0 (25.0)	0.0
Qwen3-14B OFF	mid_corrupt_stop	200	96.5	3.0	0.5 (25.0)	0.0
Qwen3-14B OFF	mid_corrupt_full	200	0.0	100.0	0.0 (25.0)	0.0
Qwen3-14B ON	corrupted_expert	200	19.5	77.5	3.0 (25.0)	0.0
Qwen3-14B ON	wrong_strategy_corrupted	200	13.0	84.0	3.0 (25.0)	0.0
Qwen3-14B ON	mid_corrupt_stop	200	34.5	64.5	0.0 (25.0)	1.0
Qwen3-14B ON	mid_corrupt_full	200	33.0	66.5	0.5 (25.0)	0.0

Table 66: Outcome breakdown at the hardest tier, date.

model	obj11 CE	obj11 mid	obj11 worst adv	obj11 WSC gold	nav20 CE	nav20 mid	nav20 worst adv	nav20 WSC gold	s3 CE	s3 mid	s3 worst adv	s3 WSC gold
Gemma-2-9B	61.0	not run	99.5 (187) skip	95.0	70.5	not run	78.3 (60) last_step	56.5†	99.5	not run	-33.0 pts skip (states 100%)	99.5
Llama-3.1-8B	92.5	not run	43.9 (187) skip	60.5	30.0†	not run	95.0 (60) last_step	0.0†	9.5†	not run	-8.5 pts skip (states 100%)	30.5†
Mistral-Small-22B	91.0	not run	76.5 (187) skip	94.0	100.0	not run	93.3 (60) last_step	96.0	87.5	not run	-14.5 pts skip (states 100%)	79.0
Qwen-2.5-14B	89.0	not run	46.2 (156) parity	89.0	97.5	not run	63.3 (60) last_step	88.0	71.0	not run	-12.5 pts skip (states 12%)	97.5
Qwen3-14B OFF	98.5	90.5	72.4 (156) parity	99.5	99.5	60.4 (101)	100.0 (60) last_step	100.0	98.5	96.5	-8.5 pts skip (states 62%)	100.0
Qwen3-14B ON	38.5	40.5	24.4 (156) parity	99.0	32.0	13.9 (101)	16.7 (60) last_step	100.0	19.5	34.5	-15.5 pts wrong_direction (states 2%)	98.5

Table 67: Cross-task summary at the hardest tier. *CE*: adoption on the corrupted solution; *mid*: adoption on the corrupted mid-step, cut (Qwen3 only); *worst adv*: adoption of the most-adopted deceptive strategy, with n and the rule; *WSC gold*: accuracy with the wrong strategy in front of the correct solution. Date deceptive strategies are given as accuracy lost.

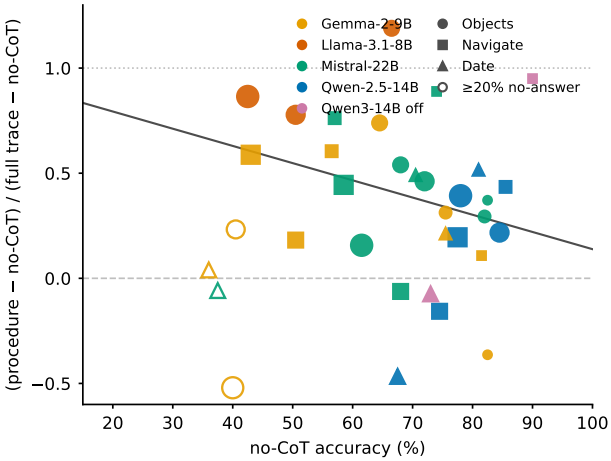


Figure 2: **How much of the gain from the full worked solution a correct strategy captures on its own.** $y = (\mathrm{acc}(\texttt{correct_strategy}) - \mathrm{acc}(\texttt{no_cot}))/(\mathrm{acc}(\texttt{expert_stripped}) - \mathrm{acc}(\texttt{no_cot}))$, one point per model and tier, shape by task, only cells with at least 10 points of headroom (the ratio is unstable below that). The line is the fit through the cells without a no-answer problem; the trend is carried by objects alone.

Task	Condition	Slope	Intercept	R^2	Cells	95% CI (problems)	95% CI (cells)
Objects	correct strategy	0.43	52.5	0.66	23	[0.36, 0.51]	[0.32, 0.59]
	25% solution	0.46	51.7	0.60	25	[0.39, 0.51]	[0.29, 0.62]
	50% solution	0.38	59.3	0.56	25	[0.32, 0.43]	[0.21, 0.52]
	75% solution	0.33	66.0	0.57	25	[0.27, 0.38]	[0.18, 0.47]
	100% solution	0.33	67.5	0.61	24	[0.27, 0.38]	[0.24, 0.39]
Navigate	correct strategy	0.56	39.3	0.56	20	[0.47, 0.64]	[0.34, 0.85]
	25% solution	0.66	29.0	0.61	20	[0.57, 0.74]	[0.39, 0.98]
	50% solution	0.66	32.3	0.63	19	[0.56, 0.74]	[0.39, 0.99]
	75% solution	0.49	48.5	0.52	19	[0.40, 0.57]	[0.29, 0.76]
	100% solution	0.03	95.8	0.03	16	[0.01, 0.07]	[−0.04, 0.16]
Date	correct strategy	0.98	4.1	0.82	12	[0.83, 1.10]	[0.69, 1.37]
	25% solution	0.96	5.2	0.91	12	[0.82, 1.07]	[0.77, 1.13]
	50% solution	0.23	70.8	0.14	12	[0.12, 0.33]	[−0.12, 0.59]
	75% solution	0.20	78.7	0.51	12	[0.12, 0.28]	[0.04, 0.30]
	100% solution	0.07	93.3	0.37	10	[0.03, 0.11]	[0.01, 0.12]

Table 68: Ordinary least-squares fits of accuracy under each condition on capability, pooled over the five instruct models and all tiers of the scenario; the slopes are those of Table 4. The intercept is in accuracy points. *Cells* is the number of model–tier cells in the fit. The two 95% confidence intervals are bootstrapped, resampling problems within each cell and resampling cells, respectively.

D CORRUPTED SOLUTIONS: ADDITIONAL TABLES

This appendix gives every tier behind two claims of Section 5: that a wrong middle step is carried forward (Section D.1) and that the copy of a wrong final value is silent (Section D.2). Which planted errors are caught, and by which model, is the subject of Appendix E, which precedes this one.

D.1 A WRONG STEP IN THE MIDDLE

The mid-derivation conditions (Section 5) exist for Qwen3-14B in both modes; the body reports thinking off, and thinking on is discussed in Section 8. Table 69 is the full version of Table 6 in the body, at every tier. The three tables after it give accuracy under the three conditions at every tier, off and on; Table 73 their difference; Table 74 the navigate problems on which the miscount does not change the answer, kept as a null control; Table 75 what the generations look like; and Table 76 detection language crossed with outcome, which is the source of the detection rates quoted in Section 5 (22–68% at the easy tiers, 0–8% at the hardest).

	Objects, N					Navigate, steps				Date		
What is in the prompt	3	5	7	9	11	5	10	15	20	s_1	s_2	s_3
% correct												
cut at a correct step k (control)	100.0	100.0	100.0	100.0	97.0	100.0	99.0	88.1	92.1	99.5	98.5	100.0
cut at a wrong step k	52.5	17.0	8.5	8.5	7.5	45.9	46.5	21.8	39.6	78.0	63.0	3.0
wrong step k, then a correct ending	100.0	100.0	100.0	98.5	99.0	71.8	97.0	99.0	99.0	100.0	100.0	100.0
% giving the answer the wrong step leads to												
cut at a wrong step k	47.5	82.0	91.0	91.5	90.5	54.1	51.5	78.2	60.4	21.0	33.5	96.5

Table 69: **A wrong middle step is carried forward, at every tier.** Qwen3-14B with thinking off. Cutting the worked solution at a correct step leaves the model to finish correctly; cutting it at a wrong step makes it finish from that step; a wrong step followed by correct steps and a correct ending costs nothing. Navigate cells are the 85–101 problems per tier on which the miscount changes the answer.

	mid_clean_stop					mid_corrupt_stop					mid_corrupt_full				
	obj3	obj5	obj7	obj9	obj11	obj3	obj5	obj7	obj9	obj11	obj3	obj5	obj7	obj9	obj11
OFF	100.0	100.0	100.0	100.0	97.0	52.5	17.0	8.5	8.5	7.5	100.0	100.0	100.0	98.5	99.0
ON	100.0	100.0	100.0	98.0	99.0	99.5	97.5	78.0	65.0	57.0	99.5	98.5	99.5	97.5	99.5

Table 70: Landmine conditions on objects, Qwen3-14B off and on.

	mid_clean_stop				mid_corrupt_stop				mid_corrupt_full			
	nav5	nav10	nav15	nav20	nav5	nav10	nav15	nav20	nav5	nav10	nav15	nav20
OFF	100.0	99.0	88.1	92.1	45.9	46.5	21.8	39.6	71.8	97.0	99.0	99.0
ON	100.0	100.0	89.1	91.1	94.1	94.9	92.1	86.1	100.0	97.0	98.0	98.0

Table 71: Landmine conditions on navigate, answer-changing landmines only.

D.2 THE COPY IS SILENT

The tables below give every tier, for the correct worked solution (`expert_stripped`) and the corrupted one (`corrupted_expert`): the median length of the generation on the correct worked solution and on the corrupted one, the share of generations that retrace at least one step on the correct solution (the number in parentheses is the stricter reading that requires a full retrace), and detection language crossed with outcome on the corrupted solution at the two hardest tiers. Llama's navigate and date cells are dominated by empty generations (d is the count).

	mid_clean_stop			mid_corrupt_stop			mid_corrupt_full		
	s1	s2	s3	s1	s2	s3	s1	s2	s3
OFF	99.5	98.5	100.0	78.0	63.0	3.0	100.0	100.0	100.0
ON	100.0	97.5	94.0	98.0	94.5	64.5	99.0	94.0	66.5

Table 72: Landmine conditions on date.

	obj3	obj5	obj7	obj9	obj11	nav5	nav10	nav15	nav20	s1	s2	s3
OFF	−47.5	−83.0	−91.5	−91.5	−89.5	−54.1	−52.5	−66.3	−52.5	−21.5	−35.5	−97.0
ON	−0.5	−2.5	−22.0	−33.0	−42.0	−5.9	−5.1	+3.0	−5.0	−2.0	−3.0	−29.5

Table 73: Landmine effect: `mid_corrupt_stop` − `mid_clean_stop`.

model	nav5	nav10	nav15	nav20
Qwen3-14B OFF	99.0 (99)	99.0 (100)	99.0 (99)	100.0 (99)
Qwen3-14B ON	100.0 (99)	100.0 (100)	100.0 (99)	100.0 (99)

Table 74: Navigate landmine null control: landmines that do not change the Yes/No answer.

Table 75: Mid-derivation corruption, Qwen3-14B, thinking off and on. Each cell reads detection-language rate / median generation length in characters (for thinking-on: total, with thought+answer in parentheses) / re-derivation rate. `clean_stop` cuts the correct trace at a middle step; `corrupt_stop` corrupts that step and cuts; `corrupt_full` corrupts it and carries the error through to a stated final answer that is nevertheless correct. Navigate is restricted to answer-changing landmines. The keyword lists are the author-reviewed ones of the detection analysis; under thinking-on the same phrases occur on clean traces (12–70%), so for that mode the adoption-bin table is the reliable read.

		thinking off			thinking on		
Task	Size	clean stop	corrupt stop	corrupt full	clean stop	corrupt stop	corrupt full
	obj3	0% / 231 / 100%	22% / 359 / 100%	0% / 123 / 0%	70% / 1171 (179+943) / 100%	86% / 1601 (594+1002) / 100%	68% / 1154 (87+984) / 100%
	obj5	0% / 525 / 100%	11% / 536 / 100%	0% / 174 / 0%	42% / 1704 (463+1214) / 100%	82% / 2287 (894+1354) / 100%	48% / 1434 (79+1287) / 72%
	obj7	0% / 835 / 100%	6% / 847 / 100%	0% / 199 / 0%	30% / 2130 (767+1320) / 100%	72% / 2691 (1136+1382) / 100%	32% / 1500 (82+1380) / 44%
	obj9	0% / 1283 / 100%	5% / 1290 / 100%	2% / 114 / 0%	12% / 2694 (1214+1439) / 100%	61% / 3041 (1551+1450) / 100%	20% / 1539 (77+1443) / 48%
	obj11	1% / 1842 / 100%	4% / 1837 / 100%	0% / 102 / 0%	22% / 3445 (1935+1570) / 100%	61% / 3709 (2130+1585) / 100%	28% / 1759 (86+1596) / 52%
	nav5	1% / 145 / 82%	20% / 154 / 69%	0% / 44 / 1%	21% / 867 (151+701) / 99%	58% / 968 (219+686) / 95%	25% / 467 (42+412) / 100%
	nav10	1% / 286 / 98%	36% / 411 / 95%	0% / 61 / 21%	26% / 1286 (315+991) / 99%	78% / 1721 (648+1028) / 97%	38% / 938 (146+672) / 91%
	nav15	3% / 334 / 100%	21% / 335 / 100%	0% / 44 / 0%	19% / 1583 (357+1199) / 100%	57% / 1756 (414+1235) / 100%	3% / 759 (50+697) / 50%
	nav20	1% / 500 / 100%	37% / 767 / 97%	3% / 62 / 18%	24% / 1841 (512+1318) / 99%	83% / 2424 (1010+1436) / 100%	32% / 1356 (138+1178) / 76%
	s1	28% / 450 / 41%	68% / 534 / 70%	3% / 97 / 0%	54% / 1283 (436+834) / 64%	92% / 1457 (629+827) / 74%	22% / 768 (74+686) / 65%
	s2	20% / 405 / 90%	62% / 490 / 96%	3% / 76 / 0%	44% / 1251 (357+864) / 99%	92% / 1710 (825+925) / 98%	22% / 846 (72+782) / 98%
	s3	6% / 112 / 41%	8% / 126 / 99%	0% / 31 / 0%	52% / 1328 (145+1070) / 100%	66% / 1880 (644+1154) / 100%	34% / 1159 (43+1106) / 100%

Table 76: Detection language crossed with outcome on `mid_corrupt_stop`, Qwen3-14B. "Implied" is the answer obtained by carrying the planted step forward correctly. When the thinking-off model verbalizes noticing, it recovers ("noticed & implied" is 0–8%); when it does not, it carries the planted step forward, on 90% of rows at $N = 11$ and on 89% at date s3. Percentages of rows; residual rows are "other" or no answer.

Task	Size	Mode	n	detected	noticed & gold	noticed & implied	silent & implied	silent & gold
objects	obj3	off	200	22%	22%	0%	48%	31%
	obj3	on	200	86%	86%	0%	0%	13%
	obj5	off	200	11%	10%	0%	82%	6%
	obj5	on	200	82%	82%	1%	1%	16%
	obj7	off	200	6%	6%	0%	90%	3%
	obj7	on	200	72%	66%	6%	16%	12%
	obj9	off	200	5%	5%	0%	92%	4%
	obj9	on	200	61%	56%	4%	28%	10%
	obj11	off	200	4%	3%	0%	90%	4%
	obj11	on	200	61%	53%	7%	34%	4%
navigate	nav5	off	85	20%	19%	1%	53%	27%
	nav5	on	85	58%	52%	6%	0%	42%
	nav10	off	99	36%	35%	1%	51%	11%
	nav10	on	99	78%	75%	3%	2%	20%
	nav15	off	101	21%	15%	6%	72%	7%
	nav15	on	101	57%	53%	4%	4%	39%
	nav20	off	101	37%	30%	7%	53%	10%
	nav20	on	101	83%	73%	10%	4%	13%
date	s1	off	200	68%	61%	6%	14%	17%
	s1	on	200	92%	90%	1%	0%	8%
	s2	off	200	62%	53%	6%	28%	10%
	s2	on	200	92%	89%	4%	2%	6%
	s3	off	200	8%	0%	8%	89%	2%
	s3	on	200	66%	52%	14%	21%	13%

model	obj3	obj5	obj7	obj9	obj11
Gemma-2-9B	74 → 70	65 → 68	74 → 73	74 → 74	75 → 75
Llama-3.1-8B	93 → 95	83 → 68	83 → 55	84 → 56	76 → 56
Mistral-Small-22B	52 → 53	56 → 54	54 → 56	57 → 58	55 → 57
Qwen-2.5-14B	101 → 107	97 → 101	99 → 101	99 → 101	103 → 101
Qwen3-14B OFF	160 → 129	158 → 175	164 → 206	114 → 198	95 → 104
Qwen3-14B ON	105 → 1446	106 → 1649	100 → 1890	106 → 1664	110 → 2064

Table 77: Median generation length, correct vs corrupted trace, objects.

model	nav5	nav10	nav15	nav20
Gemma-2-9B	47 → 47	54 → 53	49 → 51	53 → 47
Llama-3.1-8B	100 → 56	56 → 96	96 → 93	60 → 89
Mistral-Small-22B	59 → 52	60 → 52	58 → 58	44 → 48
Qwen-2.5-14B	67 → 67	79 → 67	67 → 67	81 → 76
Qwen3-14B OFF	44 → 48	60 → 49	47 → 49	61 → 53
Qwen3-14B ON	734 → 866	970 → 1117	1197 → 1338	1469 → 1444

Table 78: Median generation length, correct vs corrupted trace, navigate.

model	s1	s2	s3
Gemma-2-9B	36 → 37	31 → 37	30 → 30
Llama-3.1-8B	40 → 40	40 → 40	40 → 40
Mistral-Small-22B	44 → 67	74 → 68	52 → 52
Qwen-2.5-14B	50 → 278	50 → 148	51 → 51
Qwen3-14B OFF	118 → 97	103 → 79	31 → 31
Qwen3-14B ON	738 → 853	842 → 1020	1115 → 1274

Table 79: Median generation length, correct vs corrupted trace, date.

model	obj3	obj5	obj7	obj9	obj11
Gemma-2-9B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Llama-3.1-8B	10.5 (1.5)	9.0 (1.0)	5.5 (0.0)	4.0 (1.5)	3.0 (0.0)
Mistral-Small-22B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen-2.5-14B	2.0 (0.5)	3.5 (0.5)	0.5 (0.0)	1.0 (0.0)	1.5 (0.0)
Qwen3-14B OFF	30.0 (1.0)	12.0 (1.5)	2.5 (0.5)	0.0 (0.0)	0.5 (0.0)
Qwen3-14B ON	3.5 (3.5)	3.0 (3.0)	1.0 (1.0)	1.5 (0.5)	1.0 (1.0)

Table 80: Re-derivation rate on the correct full trace, objects.

model	nav5	nav10	nav15	nav20
Gemma-2-9B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Llama-3.1-8B	0.0 (0.0) [d=86]	0.0 (0.0) [d=53]	0.0 (0.0) [d=95]	0.0 (0.0) [d=72]
Mistral-Small-22B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen-2.5-14B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen3-14B OFF	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen3-14B ON	94.0 (90.0)	78.0 (77.0)	61.0 (61.0)	76.5 (76.5)

Table 81: Re-derivation rate on the correct full trace, navigate.

model	s1	s2	s3
Gemma-2-9B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Llama-3.1-8B	0.0 (0.0) [d=75]	0.0 (0.0) [d=126]	0.0 (0.0) [d=177]
Mistral-Small-22B	0.0 (0.0)	1.0 (0.0)	3.3 (0.0) [d=17]
Qwen-2.5-14B	0.0 (0.0)	4.5 (3.0)	0.0 (0.0)
Qwen3-14B OFF	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen3-14B ON	66.0 (45.0)	99.5 (89.0)	100.0 (100.0)

Table 82: Re-derivation rate on the correct full trace, date.

tier	model	n	detected	noticed & recovered	noticed & followed	silent copy	silent gold	FP on expert_stripped
obj9	Gemma-2-9B	200	0.0	0.0	0.0	68.5	8.5	0.0
obj9	Llama-3.1-8B	200	0.0	0.0	0.0	95.5	1.5	3.0
obj9	Mistral-Small-22B	200	0.0	0.0	0.0	94.5	5.5	0.0
obj9	Qwen-2.5-14B	200	4.0	0.5	1.0	92.5	1.5	1.5
obj9	Qwen3-14B OFF	200	0.0	0.0	0.0	99.5	0.0	0.5
obj9	Qwen3-14B ON	200	42.0	38.0	4.0	34.5	22.5	1.0
obj11	Gemma-2-9B	200	0.0	0.0	0.0	61.0	8.5	0.0
obj11	Llama-3.1-8B	200	1.5	0.0	1.0	91.5	3.0	3.0
obj11	Mistral-Small-22B	200	0.0	0.0	0.0	91.0	8.0	0.0
obj11	Qwen-2.5-14B	200	6.0	1.0	0.5	88.5	2.0	5.0
obj11	Qwen3-14B OFF	200	1.5	0.0	1.0	97.5	1.0	0.5
obj11	Qwen3-14B ON	200	46.0	43.0	3.0	35.5	18.5	2.5

Table 83: Detection language $\times$ outcome on the corrupted trace, objects.

tier	model	n	detected	noticed & recovered	noticed & followed	silent copy	silent gold	FP on expert_stripped
nav15	Gemma-2-9B	200	0.0	0.0	0.0	99.5	0.5	0.0
nav15	Llama-3.1-8B	200	0.0	0.0	0.0	37.0	1.5	0.0
nav15	Mistral-Small-22B	200	0.0	0.0	0.0	100.0	0.0	0.0
nav15	Qwen-2.5-14B	200	0.0	0.0	0.0	98.0	2.0	0.0
nav15	Qwen3-14B OFF	200	0.0	0.0	0.0	100.0	0.0	0.0
nav15	Qwen3-14B ON	200	32.5	29.0	3.5	15.0	52.5	0.5
nav20	Gemma-2-9B	200	0.0	0.0	0.0	70.5	29.5	0.0
nav20	Llama-3.1-8B	200	0.0	0.0	0.0	30.0	15.5	0.0
nav20	Mistral-Small-22B	200	0.0	0.0	0.0	100.0	0.0	0.0
nav20	Qwen-2.5-14B	200	0.0	0.0	0.0	97.5	2.5	2.5
nav20	Qwen3-14B OFF	200	0.0	0.0	0.0	99.5	0.0	3.0
nav20	Qwen3-14B ON	200	31.0	28.0	3.0	29.0	40.0	1.5

Table 84: Detection language $\times$ outcome on the corrupted trace, navigate.

tier	model	n	detected	noticed & recovered	noticed & followed	silent copy	silent gold	FP on expert_stripped
s3	Gemma-2-9B	200	0.0	0.0	0.0	99.5	0.5	0.0
s3	Llama-3.1-8B	200	0.0	0.0	0.0	9.5	0.0	0.0
s3	Mistral-Small-22B	200	3.0	0.0	0.5	87.0	0.5	1.0
s3	Qwen-2.5-14B	200	31.0	27.0	3.0	68.0	1.0	1.5
s3	Qwen3-14B OFF	200	1.5	1.5	0.0	98.5	0.0	0.0
s3	Qwen3-14B ON	200	51.5	40.0	9.5	10.0	37.5	10.5

Table 85: Detection language $\times$ outcome on the corrupted trace, date.

E WHICH PLANTED ERRORS ARE CAUGHT

Section 5 says that a wrong final value is adopted almost everywhere, with a few exceptions that are specific to a model and a task. This appendix holds the evidence behind that paragraph: the adoption table at every tier and adoption against capability over every cell (Section E.1), what Gemma-2's low objects cells are made of (Section E.2), which of the three shapes an objects corruption can take is caught and by whom (Section E.3), and what decides whether Qwen-2.5 adopts a planted date (Section E.4).

E.1 ADOPTION AT EVERY TIER, AGAINST CAPABILITY

Table 5 gives, for every model and tier, the share of answers equal to the planted value when the worked solution has one wrong item in its final state, next to the model's accuracy with nothing in the prompt, and Figure 3 plots one against the other. If capability protected a model from a wrong value, the cloud would slope down. It does not. Over the 53 usable cells of Table 5 (all twelve tiers of the five instruct models, minus Llama's navigate and date cells) the Spearman correlation between adoption and capability is ρ=0.16 (p=0.24), with the sign pointing the wrong way for the hypothesis; within a single model the correlations range from -0.39 (Qwen3-off) to 0.26 (Gemma-2), none of them significant on twelve tiers. Leaving out Qwen-2.5's three date cells, the one exception with an explanation of its own, changes nothing (ρ=0.22, p=0.13).

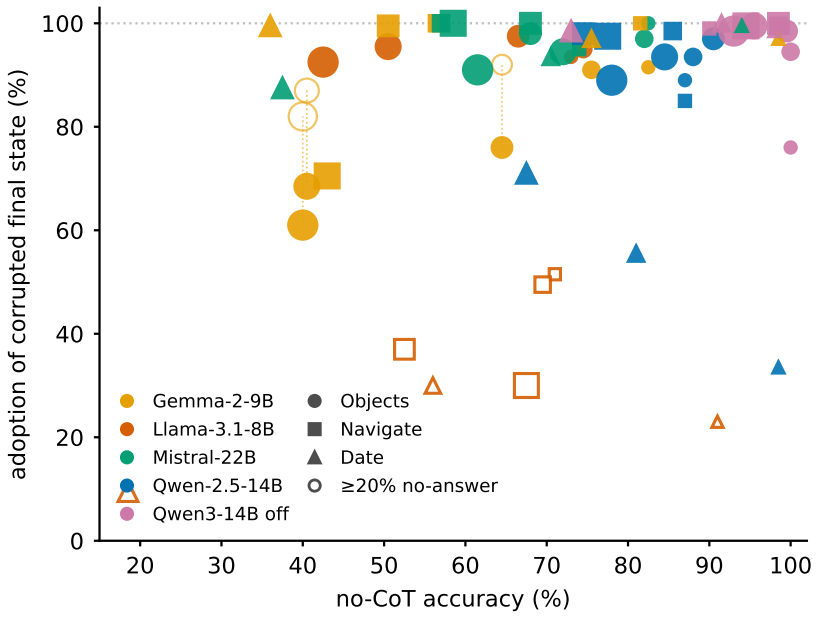


Figure 3: **Adoption of the planted final value against capability.** One dot per model and tier; the x-axis is the model's accuracy with nothing in the prompt, the y-axis the share of answers equal to the planted value when the worked solution has one wrong item in its final state. Cells dominated by empty generations are hollow. The three low Qwen-2.5 dots are its date tiers (Section E.4); the low Gemma-2 dots are its objects tiers at N=9, 11 (Section E.2).

E.2 GEMMA-2 ON OBJECTS: THE PLANTED ITEM UNDER ANOTHER LETTER

Gemma-2 is the only model whose adoption on objects drops with N, to 68% at N=9 and 61% at N=11 under the letter-based extraction used everywhere else in the paper. Most of the drop is not resistance. Table 86 rescores the same generations by the item the answer names instead of the option letter it emits. The two readings disagree on 20% of rows at N=9 and 22% at N=11, and the disagreements run one way: 32 of 40 and 38 of 44 are answers that name the planted item and label it with a different option's letter. Scored by content, adoption is 87% and 82%, and the share

of answers that land on the correct item is 7.5% under either scoring. The rows on which Gemma-2 mislabels are not visibly different from the rest: median generation length is 72–76 characters in every bin, no row retraces a step, every named item is among the options and none is a freebie. The mislabelling is less frequent when the queried person is in the final swap (8.6% against 24.8% at N=11, Fisher p=0.042; 16.0% against 21.3% at N=9, n.s.), which is also where content-scored adoption is highest (94% against 79% at N=11, p=0.05; 96% against 84% at N=9, p=0.03). We do not build on this beyond Section 5: on the other four models the two scorings agree, and on navigate, a yes/no task, there is no third answer for Gemma-2 to give. The mislabelling is not created by the corruption: on the correct worked solution the two readings already disagree on 9.5% of rows at N=9 and 15.5% at N=11, against 4.5% and 2.5% with nothing in the prompt, and the corruption raises it to 20.0% and 22.0%.

Tier	Scoring	planted item	other item	correct item	letter ≠ item
N=9	option letter	137 (68.5%)	46 (23.0%)	17 (8.5%)	40 (20.0%)
	item named	174 (87.0%)	11 (5.5%)	15 (7.5%)	
N=11	option letter	122 (61.0%)	61 (30.5%)	17 (8.5%)	44 (22.0%)
	item named	164 (82.0%)	21 (10.5%)	15 (7.5%)	

Table 86: **Gemma-2's answers on the corrupted solution, scored two ways.** Counts of 200 problems whose answer is the planted item, another item, or the correct one, reading the option letter the generation emits or the item it names. The last column counts rows on which the two readings disagree; 32 of 40 and 38 of 44 of them name the planted item.

E.3 Objects: the three shapes a corruption can take

On objects our corruption gives the queried person another person's item and that person the queried person's, with the other person drawn uniformly at random. The draw decides how the corrupted list reads. If it lands on the queried person's partner in the final swap, the corruption undoes that swap, and the last list is identical to the one above it: the text announces a trade and nothing beneath it changes. If it lands on one person in the final swap and one outside it, three people's items change where the sentence named two. If it lands on neither, the announced trade happens and a second, unannounced one happens as well. We call these an *undone swap*, a *three-person cycle* and an *extra exchange*. The first can be caught by comparing two adjacent lines. The other two look like the aftermath of some swap, and catching them requires knowing who was supposed to change. Which kind a problem receives is decided by the draw and by nothing else, so copy rates across the three are comparable within a model.

Table 87 pools them over N. Every instruct model copies cycles and extra exchanges at 66–100%, and none tells the two apart. Two models copy the undone swap less: Qwen3-off at 37% and Qwen-2.5 at 81%. Both differences are significant when the comparison is restricted to problems where the queried person is in the final swap, so that only the draw differs (Mantel–Haenszel stratified by N, $p < 0.001$; Table 91); Llama's 88 against 96 is marginal (p=0.02), and Gemma-2 and Mistral show no difference at all. So the only check present in this roster fires on a contradiction between two adjacent lines, and nothing that requires tracking who should have changed fires in any model. This also bounds what the check is worth. The share of each kind is set by N (Table 88): undone swaps are a third of corruptions at N=3 and 2% at N=11, so the case a model can catch is the case that stops occurring as the problem gets harder. Table 89 gives adoption by kind, model and N, with the query-only and partner-only cases of the three-person cycle kept apart, and Table 90 the same pooled over the two cycle cases.

Table 89: Adoption by kind of planted error, per model and N (% of problems, n in parentheses).

tier	model	both (swap looks undone)	query only (3-cycle)	partner only (3-cycle)	neither (extra exchange)
obj3	Gemma-2-9B	95.5 (66)	92.5 (67)	86.6 (67)	— (0)
obj3	Llama-3.1-8B	87.9 (66)	95.5 (67)	97.0 (67)	— (0)
obj3	Mistral-Small-22B	100.0 (66)	100.0 (67)	100.0 (67)	— (0)
obj3	Qwen-2.5-14B	78.8 (66)	94.0 (67)	94.0 (67)	— (0)
obj3	Qwen3-14B OFF	27.3 (66)	100.0 (67)	100.0 (67)	— (0)
obj3	Qwen3-14B ON	0.0 (66)	0.0 (67)	3.0 (67)	— (0)
obj5	Gemma-2-9B	94.4 (18)	93.9 (49)	91.9 (74)	86.4 (59)

tier	model	both (swap looks undone)	query only (3-cycle)	partner only (3-cycle)	neither (extra exchange)
obj5	Llama-3.1-8B	83.3 (18)	98.0 (49)	95.9 (74)	94.9 (59)
obj5	Mistral-Small-22B	100.0 (18)	100.0 (49)	95.9 (74)	94.9 (59)
obj5	Qwen-2.5-14B	77.8 (18)	98.0 (49)	93.2 (74)	94.9 (59)
obj5	Qwen3-14B OFF	44.4 (18)	100.0 (49)	100.0 (74)	98.3 (59)
obj5	Qwen3-14B ON	0.0 (18)	18.4 (49)	6.8 (74)	1.7 (59)
obj7	Gemma-2-9B	87.5 (8)	84.5 (58)	73.5 (49)	70.6 (85)
obj7	Llama-3.1-8B	100.0 (8)	94.8 (58)	98.0 (49)	98.8 (85)
obj7	Mistral-Small-22B	100.0 (8)	100.0 (58)	95.9 (49)	97.6 (85)
obj7	Qwen-2.5-14B	100.0 (8)	98.3 (58)	98.0 (49)	95.3 (85)
obj7	Qwen3-14B OFF	62.5 (8)	100.0 (58)	100.0 (49)	100.0 (85)
obj7	Qwen3-14B ON	0.0 (8)	37.9 (58)	28.6 (49)	5.9 (85)
obj9	Gemma-2-9B	50.0 (4)	82.6 (46)	62.9 (35)	65.2 (115)
obj9	Llama-3.1-8B	100.0 (4)	97.8 (46)	94.3 (35)	94.8 (115)
obj9	Mistral-Small-22B	100.0 (4)	100.0 (46)	100.0 (35)	90.4 (115)
obj9	Qwen-2.5-14B	100.0 (4)	95.7 (46)	97.1 (35)	91.3 (115)
obj9	Qwen3-14B OFF	75.0 (4)	100.0 (46)	100.0 (35)	100.0 (115)
obj9	Qwen3-14B ON	25.0 (4)	63.0 (46)	48.6 (35)	26.1 (115)
obj11	Gemma-2-9B	66.7 (3)	87.5 (32)	60.0 (40)	54.4 (125)
obj11	Llama-3.1-8B	66.7 (3)	100.0 (32)	92.5 (40)	91.2 (125)
obj11	Mistral-Small-22B	100.0 (3)	93.8 (32)	95.0 (40)	88.8 (125)
obj11	Qwen-2.5-14B	66.7 (3)	100.0 (32)	87.5 (40)	87.2 (125)
obj11	Qwen3-14B OFF	100.0 (3)	100.0 (32)	100.0 (40)	97.6 (125)
obj11	Qwen3-14B ON	0.0 (3)	68.8 (32)	50.0 (40)	28.0 (125)

E.4 DATE: THE PLANTED DATE IS JUDGED BY HOW FAR OFF IT IS

Every date solution has the same two steps, establish today's date and apply the shift, and the corruption always alters the second, leaving the first correct. Two consequences. First, the planted date sits a known number of days from the date the solution's own first step implies. Second, the planted line either refutes itself, when it carries both operands and a result that does not follow (*one-line*: "May 26, 2010 + 1 day = May 27, 2009"), or contradicts only the line above it, when the shift is an identity and the line simply restates today's date wrongly (*cross-step*). Only s_1 holds both kinds (92 and 108 problems; s_2 and s_3 are one-line throughout), and there the kind is perfectly confounded with how the problem states today's date: every one-line problem gives it outright, every cross-step problem gives it indirectly ("yesterday was", "the deadline is"). Four of the six models adopt the two kinds at the same rate; Llama adopts the one-line kind more (40% against 8%, Fisher $p < 10^{-7}$) and Qwen3-on marginally so (14% against 7%, p=0.10), but under the confound the natural reading is that these two lose accuracy when today's date has to be inferred, not that they check the arithmetic line. We draw nothing from this contrast.

What does vary with the planted error is how far off it is. Table 92 gives adoption by the distance between the planted date and the correct one, pooled over the three tiers; the *overall* column is the model's adoption on all date rows, and a model near 100% has no room to vary. Gemma-2, Mistral and Qwen3-off adopt 94–100% at every distance, so their flat rows are a ceiling. Among the three models with room to move, Qwen-2.5 is the one with a gradient: 76% when the planted date is one day off, down to 23% when it is a year off (Spearman $\rho = -0.36$), with the same ordering inside each of the three tiers. Llama and Qwen3-on slope the same way much more weakly (-0.13, -0.17) from a low base. By the field of the date that was altered, Qwen-2.5 adopts a changed day at 66% (n=312, median two days off), a changed month at 50% (n=28), month and day at 53% (n=139), and a changed year at 18% (n=98). The subtraction that would catch a one-day error is the same one that would catch a year, and no generation of any model writes it: the re-derivation markers of Section D.2 fire on none of these rows.

The reports say the same thing. Table 93 crosses the report flag of Section 5 with the same distance bins for Qwen-2.5 at s_3, the cell where it reports a problem on 31% of generations. A problem is reported on 6% of the prompts where the planted date is one day off, on about 21% of those two to thirty-one days off, and on 80% of those a year off. Year-off rows are reported at 75% against 20% for every other size of error (Fisher p=1.2×10^{-10}); a changed year is reported 86% of the time and a changed day 18%. The reports concentrate on year-off rows without being confined to them: those rows are 20% of the benchmark and supply 30 of the 62 reports. So what the report says is that the planted date looks wrong, and a date starts to look wrong at about a year.

Two things this data cannot separate. A year-sized error always changes the year field, so the size of the error and the field it lands in are the same variable here; there is no 200-day error that keeps

Model	undone swap n=99	three-person cycle n=517	extra exchange n=384
Gemma-2-9B	92	83	66
Llama-3.1-8B	88	96	95
Mistral-22B	100	98	92
Qwen-2.5-14B	81	95	91
Qwen3-14B off	37	100	99
Qwen3-14B on	1	27	18

Table 87: **Only the corruption that contradicts the adjacent line is caught.** Share of answers equal to the planted value, by the kind of corruption the random draw produced, pooled over N. The five instruct models are above the rule; Qwen3-14B with thinking on is below it (Section 8).

	N=3	N=5	N=7	N=9	N=11
undone swap / cycle / extra exchange (%)	33 / 67 / 0	9 / 62 / 30	4 / 54 / 42	2 / 40 / 58	2 / 36 / 62

Table 88: Share of the 200 problems per tier that receive each kind of corruption. At N=3 every person is in the final swap, so no extra exchange is possible.

model	obj3	obj5	obj7	obj9	obj11
Gemma-2-9B	95.5 / 89.6 / —	94.4 / 92.7 / 86.4	87.5 / 79.4 / 70.6	50.0 / 74.1 / 65.2	66.7 / 72.2 / 54.4
Llama-3.1-8B	87.9 / 96.3 / —	83.3 / 96.7 / 94.9	100.0 / 96.3 / 98.8	100.0 / 96.3 / 94.8	66.7 / 95.8 / 91.2
Mistral-Small-22B	100.0 / 100.0 / —	100.0 / 97.6 / 94.9	100.0 / 98.1 / 97.6	100.0 / 100.0 / 90.4	100.0 / 94.4 / 88.8
Qwen-2.5-14B	78.8 / 94.0 / —	77.8 / 95.1 / 94.9	100.0 / 98.1 / 95.3	100.0 / 96.3 / 91.3	66.7 / 93.1 / 87.2
Qwen3-14B OFF	27.3 / 100.0 / —	44.4 / 100.0 / 98.3	62.5 / 100.0 / 100.0	75.0 / 100.0 / 100.0	100.0 / 100.0 / 97.6
Qwen3-14B ON	0.0 / 1.5 / —	0.0 / 11.4 / 1.7	0.0 / 33.6 / 5.9	25.0 / 56.8 / 26.1	0.0 / 58.3 / 28.0

Table 90: Copy rate by kind (undone swap / three-person cycle / extra exchange), per model and N; pooled in Table 87.

model	contrast	copy rate, first kind	copy rate, cycle	MH odds ratio	p (CMH)	per N: copies first vs cycle (Fisher p)
Gemma-2-9B	q in last swap: undone vs cycle	91.9 (91/99)	88.5 (223/252)	0.90	0.994	obj3: 63/66 vs 62/67 (p 0.718); obj5: 17/18 vs 46/49 (p 1.000); obj7: 7/8 vs 49/58 (p 1.000); obj9: 2/4 vs 38/46 (p 0.174); obj11: 2/3 vs 28/32 (p 0.380)
Gemma-2-9B	q outside: exchange vs cycle	66.1 (254/384)	75.8 (150/198)	0.85	0.500	obj5: 51/59 vs 68/74 (p 0.397); obj7: 60/85 vs 36/49 (p 0.843); obj9: 75/115 vs 22/35 (p 0.841); obj11: 68/125 vs 24/40 (p 0.586)
Llama-3.1-8B	q in last swap: undone vs cycle	87.9 (87/99)	96.8 (244/252)	0.29	0.022	obj3: 58/66 vs 64/67 (p 0.128); obj5: 15/18 vs 48/49 (p 0.056); obj7: 8/8 vs 55/58 (p 1.000); obj9: 4/4 vs 45/46 (p 1.000); obj11: 2/3 vs 32/32 (p 0.086)
Llama-3.1-8B	q outside: exchange vs cycle	94.5 (363/384)	95.5 (189/198)	0.94	0.943	obj5: 56/59 vs 71/74 (p 1.000); obj7: 84/85 vs 48/49 (p 1.000); obj9: 109/115 vs 33/35 (p 1.000); obj11: 114/125 vs 37/40 (p 1.000)
Mistral-Small-22B	q in last swap: undone vs cycle	100.0 (99/99)	99.2 (250/252)	inf	0.400	obj3: 66/66 vs 67/67 (p 1.000); obj5: 18/18 vs 49/49 (p 1.000); obj7: 8/8 vs 58/58 (p 1.000); obj9: 4/4 vs 46/46 (p 1.000); obj11: 3/3 vs 30/32 (p 1.000)
Mistral-Small-22B	q outside: exchange vs cycle	92.2 (354/384)	96.5 (191/198)	0.48	0.152	obj5: 56/59 vs 71/74 (p 1.000); obj7: 83/85 vs 47/49 (p 0.623); obj9: 104/115 vs 35/35 (p 0.068); obj11: 111/125 vs 38/40 (p 0.362)
Qwen-2.5-14B	q in last swap: undone vs cycle	80.8 (80/99)	96.8 (244/252)	0.20	< 0.001	obj3: 52/66 vs 63/67 (p 0.012); obj5: 14/18 vs 48/49 (p 0.016); obj7: 8/8 vs 57/58 (p 1.000); obj9: 4/4 vs 44/46 (p 1.000); obj11: 2/3 vs 32/32 (p 0.086)
Qwen-2.5-14B	q outside: exchange vs cycle	91.4 (351/384)	93.9 (186/198)	0.78	0.616	obj5: 56/59 vs 69/74 (p 1.000); obj7: 81/85 vs 48/49 (p 0.652); obj9: 105/115 vs 34/35 (p 0.459); obj11: 109/125 vs 35/40 (p 1.000)
Qwen3-14B OFF	q in last swap: undone vs cycle	37.4 (37/99)	100.0 (252/252)	0.00	< 0.001	obj3: 18/66 vs 67/67 (p < 0.001); obj5: 8/18 vs 49/49 (p < 0.001); obj7: 5/8 vs 58/58 (p 0.001); obj9: 3/4 vs 46/46 (p 0.080); obj11: 3/3 vs 32/32 (p 1.000)
Qwen3-14B OFF	q outside: exchange vs cycle	99.0 (380/384)	100.0 (198/198)	0.00	0.378	obj5: 58/59 vs 74/74 (p 0.444); obj7: 85/85 vs 49/49 (p 1.000); obj9: 115/115 vs 35/35 (p 1.000); obj11: 122/125 vs 40/40 (p 1.000)
Qwen3-14B ON	q in last swap: undone vs cycle	1.0 (1/99)	32.5 (82/252)	0.04	< 0.001	obj3: 0/66 vs 0/67 (p 1.000); obj5: 0/18 vs 9/49 (p 0.099); obj7: 0/8 vs 22/58 (p 0.045); obj9: 1/4 vs 29/46 (p 0.289); obj11: 0/3 vs 22/32 (p 0.044)
Qwen3-14B ON	q outside: exchange vs cycle	18.5 (71/384)	28.3 (56/198)	0.31	< 0.001	obj5: 1/59 vs 5/74 (p 0.226); obj7: 5/85 vs 14/49 (p < 0.001); obj9: 30/115 vs 17/35 (p 0.021); obj11: 35/125 vs 20/40 (p 0.013)

Table 91: Randomized contrasts, Mantel–Haenszel over N. The first contrast compares the undone swap with the cycle among problems where the queried person is in the last swap; the second compares the extra exchange with the cycle among problems where it is not. In each, only the random draw differs.

Model	overall	1 day	2–7 d	8–31 d	32–366 d	Spearman ρ
Gemma-2-9B	97.8	96.7	100.0	98.0	94.7	−0.09
Llama-3.1-8B†	20.8	29.3	24.9	18.0	12.3	−0.13
Mistral-22B	93.5	95.7	93.7	94.1	90.4	−0.06
Qwen-2.5-14B	53.3	76.1	64.0	50.2	22.8	−0.36
Qwen3-14B off	99.0	100.0	100.0	99.0	96.5	−0.11
Qwen3-14B on	13.7	23.9	16.4	12.7	2.6	−0.17

Table 92: **Adoption of a planted date by its distance from the correct one**, in percent, pooled over the three date tiers. Qwen-2.5 is the only model with room to move that moves; the same ordering holds inside each tier. † Llama's date cells are excluded from claims (Section 3).

Distance	n	reports a problem	adopts
1 day	32	6.2	90.6
2–7 d	51	21.6	80.4
8–31 d	76	21.1	82.9
32–366 d	41	80.5	22.0

Table 93: **What Qwen-2.5 reports at s_3, by distance**, in percent of rows. Year-off rows are reported at 75.0% against 20.0% for every other size of error; a changed year is reported 85.7% of the time and a changed day 17.8%.

the year. And a year-off date is also the one whose year is a token the prompt does not otherwise contain: the planted year appears in the scenario on 20% of year-off rows against 56% of the others, and on 36 of the 40 year-off problems at s_3 it occurs exactly once in the whole prompt, in the option that is the planted answer. A lexical tell is therefore available, and we cannot say whether it is the one being used. Novelty and being year-off are nearly the same variable (every novel-year row is year-off, 72% of year-off rows are novel). The one within-group contrast, year-off rows split by whether the year also appears in the scenario, gives 24 of 29 reported when it does not against 6 of 11 when it does (p=0.1), which leans toward the lexical reading on too few rows to settle it. Separating the two needs a corruption that is implausible but lexically present, or novel but plausible, and this benchmark contains neither. The gradient is one model's, and we do not state it as a property of the roster.

F STRATEGIES: ADDITIONAL FIGURES AND TABLES

This appendix completes Section 6. It gives the per-cell view of what a wrong strategy does (Section F.1), the adoption of every deceptive strategy against capability and against tier, with the controls that say how often a rule's answer is produced without the rule (Section F.2), and the full stating-versus-executing tables (Section F.3). Condition names are those of Table 13.

F.1 A STRATEGY FOR THE WRONG TASK

Figure 4 plots accuracy with the other task's strategy in the prompt (`wrong_strategy`) against accuracy with nothing added; the damage figures and correlations of Table 7 are read off these points, and the accuracy under the condition at every tier is in Appendix C.1. Table 94 gives the change in outcome when the same strategy is placed in front of a worked solution, correct or corrupted. For most models and tiers the change is within a few points, which is what Section 6 reports; the exceptions are Gemma-2 on navigate and Llama on objects, where the strategy in front of a correct solution costs 18–44 points, and they are marked.

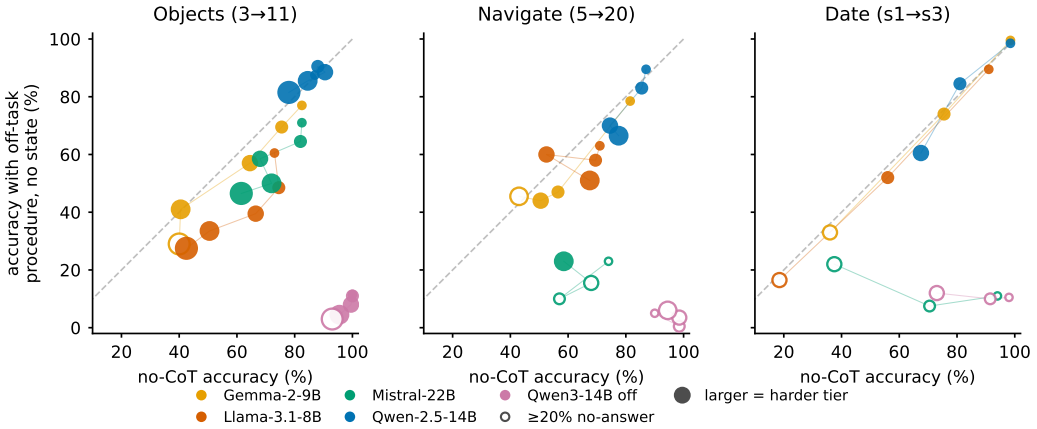


Figure 4: **A strategy for a different task, alone in the prompt, is ignored, hedged, or executed, by model.** x is accuracy with nothing added, y accuracy with the other task's strategy added (`wrong_strategy`); the diagonal is "no effect"; hollow dots have ≥20% no-answer generations.

	Objects, N					Navigate, steps				Date		
Model	3	5	7	9	11	5	10	15	20	s_1	s_2	s_3
On correct state: gold(`wrong_strategy_correct`) − gold(`expert_stripped`)												
Gemma-2-9B	+1.5	+1.5	+3.5	+8.5	+18.5	−18.0	–	−41.0†	−43.5†	−0.5	−1.0	+0.5
Llama-3.1-8B	+15.0†	−2.0	−19.5	−18.5	−19.0	–	–	–	–	–	–	–
Mistral-22B	+0.0	+1.0	−1.0	−1.5	−2.5	+0.0	+0.0	+0.0	−3.0	+0.0	+0.0	−10.0
Qwen-2.5-14B	+0.0	−3.0	−2.0	−4.0	−3.0	−3.0	−2.0	−2.5	−10.0	−0.5	+0.5	+0.0
Qwen3-14B off	+0.0	−0.5	+0.0	+0.0	−0.5	+0.0	+0.0	+0.0	+0.0	+0.0	−0.5	+0.0
On corrupted state: adoption(`wrong_strategy_corrupted`) − adoption(`corrupted_expert`)												
Gemma-2-9B	+3.5	+1.0	+12.0	+9.0	+17.0	−21.5†	−37.0†	−30.5†	+14.5	+1.0	−1.0	−0.5
Llama-3.1-8B	−2.5	−8.0	−7.0	−10.5	−17.5	–	–	–	–	–	–	–
Mistral-22B	−0.5	−0.5	−1.5	−3.0	−2.0	−0.5	+0.0	+0.0	+0.0	−0.5	+3.5	−13.5†
Qwen-2.5-14B	−2.5	−1.0	−1.0	−3.0	−3.5	−4.0	−3.0	+0.0	−1.0	−3.5	+2.5	+1.5
Qwen3-14B off	+6.0	+1.5	+1.5	−0.5	−1.0	−0.5	+0.5	+0.0	+0.5	+0.5	+0.0	+1.5

Table 94: **With a solution present, a wrong strategy changes little.** Top: the change in accuracy when the wrong strategy is prepended to the correct worked solution. Bottom: the change in adoption of the planted answer when it is prepended to the corrupted solution. Save for the marked exceptions, the differences are within a few points. †: one of the two cells has ≥20% no-answer. –: one of the two cells is ≥50% empty (Llama on navigate and date, where a full trace in the prefix produces no output).

F.2 DECEPTIVE STRATEGIES

Table 8 in the body shows two rules per task at two tiers. Figure 5 plots adoption of every rule against capability and Figure 6 against tier, and the adoption tables of Appendix C.2 give every cell. Tables 95 and 96 are the control behind the claim that the rule is what puts its answer there: how often the rule's answer is produced with no strategy, with the correct one, and with the rule itself. The three date rules imply an answer outside the six options on most problems, so Tables 97–99 report them as accuracy lost against `no_cot` on the problems where the rule is wrong, together with how often the rule is restated.

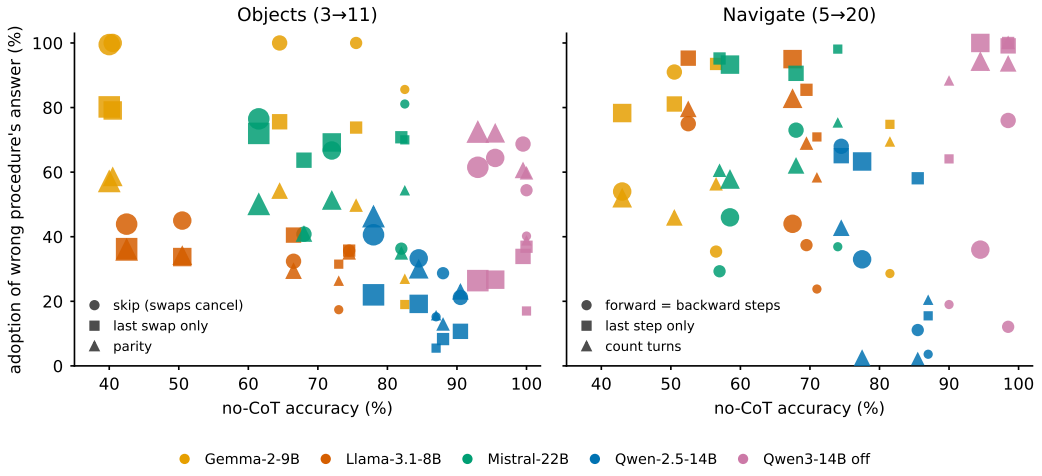


Figure 5: **Following a deceptive strategy is not ordered by capability.** y is the share of answers equal to the rule's output, on problems where that output is wrong; one dot per model, tier and rule (shape). Date rules are omitted because their output is usually not an option.

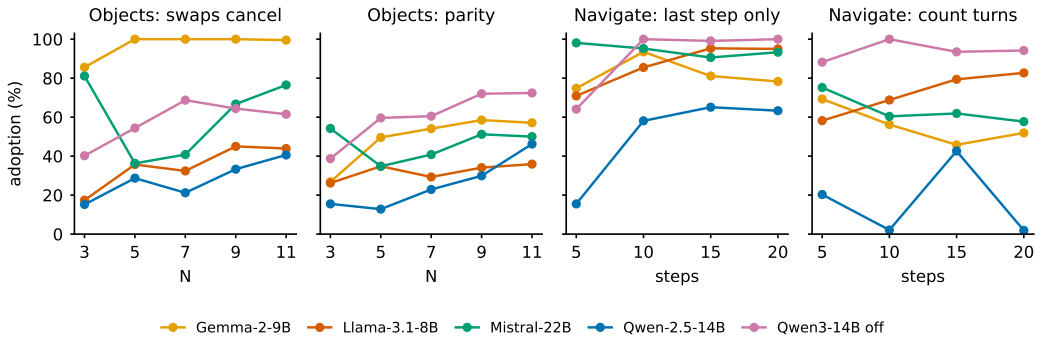


Figure 6: **Following a deceptive strategy grows with difficulty on objects and on *last step only*, and not on *count turns*.** Share of answers equal to the rule's output, by tier, for the four most-followed rules.

Table 95: Baseline-oracle control, objects. Each cell gives the share of answers equal to the rule's oracle output under no CoT / correct strategy / the rule itself, on the same non-freebie problems (n = 132–200). Without the rule the oracle answer is produced on 2–13% of problems at $N \leq 7$ and at most 23% at $N \geq 9$; with it, on 26–100%.

Model	Size	adv. skip	adv. last swap	adv. parity
Gemma-2-9B	N=3	0.09 / 0.14 / 0.86	0.12 / 0.10 / 0.19	0.06 / 0.11 / 0.27
Gemma-2-9B	N=5	0.06 / 0.09 / 1.00	0.10 / 0.10 / 0.74	0.10 / 0.10 / 0.50
Gemma-2-9B	N=7	0.08 / 0.04 / 1.00	0.13 / 0.04 / 0.76	0.12 / 0.05 / 0.54
Gemma-2-9B	N=9	0.13 / 0.07 / 1.00	0.18 / 0.06 / 0.79	0.21 / 0.07 / 0.58

Model	Size	adv. skip	adv. last swap	adv. parity
Gemma-2-9B	N=11	0.13 / 0.02 / 0.99	0.21 / 0.02 / 0.80	0.23 / 0.01 / 0.57
Llama-3.1-8B	N=3	0.07 / 0.02 / 0.17	0.16 / 0.05 / 0.32	0.10 / 0.03 / 0.26
Llama-3.1-8B	N=5	0.04 / 0.04 / 0.36	0.10 / 0.05 / 0.36	0.04 / 0.03 / 0.35
Llama-3.1-8B	N=7	0.06 / 0.03 / 0.32	0.17 / 0.06 / 0.41	0.06 / 0.03 / 0.29
Llama-3.1-8B	N=9	0.07 / 0.04 / 0.45	0.14 / 0.07 / 0.34	0.13 / 0.08 / 0.34
Llama-3.1-8B	N=11	0.13 / 0.03 / 0.44	0.21 / 0.08 / 0.36	0.14 / 0.03 / 0.36
Mistral-Small-22B	N=3	0.01 / 0.01 / 0.81	0.05 / 0.05 / 0.70	0.06 / 0.05 / 0.54
Mistral-Small-22B	N=5	0.04 / 0.02 / 0.36	0.05 / 0.02 / 0.71	0.04 / 0.06 / 0.35
Mistral-Small-22B	N=7	0.06 / 0.02 / 0.41	0.11 / 0.05 / 0.64	0.08 / 0.04 / 0.41
Mistral-Small-22B	N=9	0.04 / 0.01 / 0.67	0.07 / 0.02 / 0.69	0.07 / 0.02 / 0.51
Mistral-Small-22B	N=11	0.02 / 0.02 / 0.77	0.06 / 0.07 / 0.72	0.03 / 0.03 / 0.50
Qwen-2.5-14B	N=3	0.04 / 0.05 / 0.15	0.10 / 0.07 / 0.06	0.06 / 0.07 / 0.15
Qwen-2.5-14B	N=5	0.06 / 0.05 / 0.29	0.06 / 0.05 / 0.08	0.03 / 0.00 / 0.13
Qwen-2.5-14B	N=7	0.03 / 0.02 / 0.21	0.06 / 0.05 / 0.11	0.05 / 0.01 / 0.23
Qwen-2.5-14B	N=9	0.03 / 0.03 / 0.33	0.05 / 0.04 / 0.19	0.06 / 0.02 / 0.30
Qwen-2.5-14B	N=11	0.02 / 0.05 / 0.41	0.04 / 0.07 / 0.22	0.05 / 0.03 / 0.46
Qwen3-14B (off)	N=3	0.00 / 0.00 / 0.40	0.00 / 0.00 / 0.17	0.00 / 0.00 / 0.39
Qwen3-14B (off)	N=5	0.00 / 0.01 / 0.54	0.00 / 0.01 / 0.37	0.00 / 0.00 / 0.60
Qwen3-14B (off)	N=7	0.00 / 0.00 / 0.69	0.00 / 0.03 / 0.34	0.00 / 0.00 / 0.60
Qwen3-14B (off)	N=9	0.01 / 0.02 / 0.64	0.00 / 0.02 / 0.27	0.02 / 0.01 / 0.72
Qwen3-14B (off)	N=11	0.00 / 0.02 / 0.61	0.01 / 0.04 / 0.26	0.02 / 0.02 / 0.72
Qwen3-14B (on)	N=3	0.00 / 0.00 / 0.00	0.00 / 0.00 / 0.02	0.00 / 0.00 / 0.04
Qwen3-14B (on)	N=5	0.00 / 0.00 / 0.00	0.00 / 0.00 / 0.01	0.00 / 0.00 / 0.04
Qwen3-14B (on)	N=7	0.00 / 0.00 / 0.01	0.00 / 0.00 / 0.05	0.00 / 0.00 / 0.06
Qwen3-14B (on)	N=9	0.00 / 0.00 / 0.02	0.00 / 0.01 / 0.07	0.00 / 0.00 / 0.20
Qwen3-14B (on)	N=11	0.00 / 0.00 / 0.03	0.01 / 0.02 / 0.07	0.00 / 0.02 / 0.24

F.3 Stating a rule and executing it

Figure 7 plots, per model, rule and tier, the share of answers that restate the rule against the share that give its answer. Table 100 gives the per-model breakdown at the hardest tier that Table 9 in the body averages, and Tables 101–103 the same at every tier, with accuracy conditioned on whether the rule was stated.

Table 101: Stating a deceptive strategy vs following it, objects.

tier	rule	model	n	states % (n)	acc. if states	acc. if not	rule's answer if states	rule's answer if not
obj3	adv_skip	Gemma-2-9B	132	10.6 (14)	0.0	10.2	92.9	84.7
obj3	adv_skip	Llama-3.1-8B	132	9.1 (12)	8.3	61.7	83.3	10.8
obj3	adv_skip	Mistral-Small-22B	132	26.5 (35)	0.0	19.6	100.0	74.2
obj3	adv_skip	Qwen-2.5-14B	132	6.1 (8)	62.5	79.0	37.5	13.7
obj3	adv_skip	Qwen3-14B OFF	132	8.3 (11)	0.0	64.5	100.0	34.7
obj3	adv_skip	Qwen3-14B ON	132	11.4 (15)	100.0	100.0	0.0	0.0
obj3	adv_last_swap	Gemma-2-9B	200	2.5 (5)	0.0	65.6	100.0	16.9
obj3	adv_last_swap	Llama-3.1-8B	200	22.5 (45)	46.7	57.4	42.2	28.4
obj3	adv_last_swap	Mistral-Small-22B	200	72.0 (144)	6.2	71.4	90.3	17.9
obj3	adv_last_swap	Qwen-2.5-14B	200	6.0 (12)	41.7	89.9	16.7	4.8
obj3	adv_last_swap	Qwen3-14B OFF	200	35.5 (71)	50.7	100.0	47.9	0.0
obj3	adv_last_swap	Qwen3-14B ON	200	22.0 (44)	95.5	97.4	2.3	1.9
obj3	adv_parity	Gemma-2-9B	168	12.5 (21)	19.0	57.8	57.1	22.4
obj3	adv_parity	Llama-3.1-8B	168	58.3 (98)	10.2	57.1	32.7	17.1
obj3	adv_parity	Mistral-Small-22B	168	76.8 (129)	3.1	69.2	63.6	23.1
obj3	adv_parity	Qwen-2.5-14B	168	24.4 (41)	61.0	84.3	36.6	8.7
obj3	adv_parity	Qwen3-14B OFF	168	28.6 (48)	4.2	78.3	85.4	20.0
obj3	adv_parity	Qwen3-14B ON	168	9.5 (16)	81.2	97.4	12.5	2.6
obj11	adv_skip	Gemma-2-9B	187	1.1 (2)	0.0	0.0	100.0	99.5
obj11	adv_skip	Llama-3.1-8B	187	4.8 (9)	11.1	21.3	77.8	42.1
obj11	adv_skip	Mistral-Small-22B	187	36.9 (69)	4.3	11.9	89.9	68.6
obj11	adv_skip	Qwen-2.5-14B	187	10.7 (20)	10.0	26.3	45.0	40.1
obj11	adv_skip	Qwen3-14B OFF	187	10.2 (19)	21.1	20.2	52.6	62.5
obj11	adv_skip	Qwen3-14B ON	187	16.0 (30)	73.3	87.9	13.3	0.6
obj11	adv_last_swap	Gemma-2-9B	182	98.4 (179)	4.5	0.0	79.9	100.0
obj11	adv_last_swap	Llama-3.1-8B	182	74.7 (136)	10.3	30.4	36.0	37.0
obj11	adv_last_swap	Mistral-Small-22B	182	94.5 (172)	4.7	60.0	75.0	20.0
obj11	adv_last_swap	Qwen-2.5-14B	182	80.2 (146)	37.7	72.2	24.0	13.9
obj11	adv_last_swap	Qwen3-14B OFF	182	67.6 (123)	44.7	76.3	35.0	8.5
obj11	adv_last_swap	Qwen3-14B ON	182	36.3 (66)	62.1	87.9	15.2	1.7
obj11	adv_parity	Gemma-2-9B	156	92.3 (144)	4.2	25.0	56.2	66.7
obj11	adv_parity	Llama-3.1-8B	156	75.6 (118)	4.2	18.4	40.7	21.1
obj11	adv_parity	Mistral-Small-22B	156	89.7 (140)	5.0	0.0	50.0	50.0
obj11	adv_parity	Qwen-2.5-14B	156	68.6 (107)	6.5	38.8	58.9	18.4
obj11	adv_parity	Qwen3-14B OFF	156	46.2 (72)	6.9	17.9	76.4	69.0
obj11	adv_parity	Qwen3-14B ON	156	19.9 (31)	9.7	84.8	83.9	9.6

Table 96: Baseline-oracle control, navigate, for the two rules that bite there (no CoT / correct strategy / the rule itself, with n). Because the answer is binary, an unprompted answer matches a rule's oracle about half the time by construction, so the baseline is high; the rate with the rule is nonetheless 0.2–1.0 above it in every cell for Qwen3 (off) and in most cells for the others. The skip rule is rejected on navigate and is omitted.

Model	Size	adv. last step	adv. count turns
Gemma-2-9B	N=5	0.24 / 0.15 / 0.75 (103)	0.24 / 0.16 / 0.69 (153)
Gemma-2-9B	N=10	0.37 / 0.23 / 0.94 (62)	0.21 / 0.04 / 0.56 (48)
Gemma-2-9B	N=15	0.53 / 0.48 / 0.81 (106)	0.50 / 0.51 / 0.46 (155)
Gemma-2-9B	N=20	0.47 / 0.33 / 0.78 (60)	0.48 / 0.02 / 0.52 (52)
Llama-3.1-8B	N=5	0.37 / 0.23 / 0.71 (103)	0.31 / 0.22 / 0.58 (153)
Llama-3.1-8B	N=10	0.32 / 0.27 / 0.85 (62)	0.17 / 0.06 / 0.69 (48)
Llama-3.1-8B	N=15	0.60 / 0.53 / 0.95 (106)	0.56 / 0.55 / 0.79 (155)
Llama-3.1-8B	N=20	0.35 / 0.33 / 0.95 (60)	0.08 / 0.06 / 0.83 (52)
Mistral-Small-22B	N=5	0.18 / 0.19 / 0.98 (103)	0.33 / 0.13 / 0.75 (153)
Mistral-Small-22B	N=10	0.21 / 0.23 / 0.95 (62)	0.06 / 0.00 / 0.60 (48)
Mistral-Small-22B	N=15	0.42 / 0.47 / 0.91 (106)	0.41 / 0.44 / 0.62 (155)
Mistral-Small-22B	N=20	0.28 / 0.27 / 0.93 (60)	0.02 / 0.00 / 0.58 (52)
Qwen-2.5-14B	N=5	0.06 / 0.00 / 0.15 (103)	0.16 / 0.05 / 0.20 (153)
Qwen-2.5-14B	N=10	0.13 / 0.18 / 0.58 (62)	0.06 / 0.00 / 0.02 (48)
Qwen-2.5-14B	N=15	0.30 / 0.39 / 0.65 (106)	0.33 / 0.38 / 0.43 (155)
Qwen-2.5-14B	N=20	0.23 / 0.27 / 0.63 (60)	0.02 / 0.00 / 0.02 (52)
Qwen3-14B (off)	N=5	0.05 / 0.01 / 0.64 (103)	0.13 / 0.01 / 0.88 (153)
Qwen3-14B (off)	N=10	0.00 / 0.03 / 1.00 (62)	0.00 / 0.00 / 1.00 (48)
Qwen3-14B (off)	N=15	0.01 / 0.13 / 0.99 (106)	0.02 / 0.14 / 0.94 (155)
Qwen3-14B (off)	N=20	0.02 / 0.10 / 1.00 (60)	0.00 / 0.00 / 0.94 (52)
Qwen3-14B (on)	N=5	0.00 / 0.00 / 0.13 (103)	0.05 / 0.00 / 0.22 (153)
Qwen3-14B (on)	N=10	0.00 / 0.02 / 0.11 (62)	0.00 / 0.00 / 0.10 (48)
Qwen3-14B (on)	N=15	0.00 / 0.04 / 0.22 (106)	0.01 / 0.04 / 0.28 (155)
Qwen3-14B (on)	N=20	0.03 / 0.03 / 0.17 (60)	0.00 / 0.00 / 0.06 (52)

model	s1	s2	s3
Gemma-2-9B	98.0 → 40.8 (-57.2) · states 89%	81.0 → 34.8 (-46.2) · states 96%	36.0 → 3.0 (-33.0) · states 100%
Llama-3.1-8B	90.1 → 70.4 (-19.7) · states 100%	58.7 → 46.7 (-12.0) · states 100%	18.5 → 10.0 (-8.5) · states 100%
Mistral-Small-22B	92.1 → 79.6 (-12.5) · states 97%	73.9 → 52.7 (-21.2) · states 91%	37.5 → 23.0 (-14.5) · states 100%
Qwen-2.5-14B	98.7 → 97.4 (-1.3) · states 15%	86.4 → 88.6 (+2.2) · states 33%	67.5 → 55.0 (-12.5) · states 12%
Qwen3-14B OFF	97.4 → 93.4 (-3.9) · states 21%	94.6 → 79.9 (-14.7) · states 32%	73.0 → 64.5 (-8.5) · states 62%
Qwen3-14B ON	100.0 → 98.0 (-2.0) · states 7%	97.8 → 88.0 (-9.8) · states 16%	97.0 → 86.5 (-10.5) · states 6%

Table 97: Date `adv_skip`: accuracy lost against no-CoT on problems where the rule is wrong, and share of answers stating it.

Table 102: Stating a deceptive strategy vs following it, navigate.

tier	rule	model	n	states % (n)	acc. if states	acc. if not	rule's answer if states	rule's answer if not
nav5	adv_skip	Gemma-2-9B	84	85.7 (72)	68.1	75.0	29.2	25.0
nav5	adv_skip	Llama-3.1-8B	84	91.7 (77)	66.2	100.0	26.0	0.0
nav5	adv_skip	Mistral-Small-22B	84	88.1 (74)	62.2	70.0	37.8	30.0
nav5	adv_skip	Qwen-2.5-14B	84	67.9 (57)	96.5	96.3	3.5	3.7
nav5	adv_skip	Qwen3-14B OFF	84	84.5 (71)	81.7	76.9	18.3	23.1
nav5	adv_skip	Qwen3-14B ON	84	86.9 (73)	98.6	100.0	1.4	0.0
nav5	adv_last_step	Gemma-2-9B	103	61.2 (63)	11.1	47.5	88.9	52.5
nav5	adv_last_step	Llama-3.1-8B	103	78.6 (81)	19.8	59.1	79.0	40.9
nav5	adv_last_step	Mistral-Small-22B	103	98.1 (101)	0.0	100.0	100.0	0.0
nav5	adv_last_step	Qwen-2.5-14B	103	1.9 (2)	100.0	84.2	0.0	15.8
nav5	adv_last_step	Qwen3-14B OFF	103	58.3 (60)	6.7	76.7	93.3	23.3
nav5	adv_last_step	Qwen3-14B ON	103	68.9 (71)	85.9	87.5	12.7	12.5
nav5	adv_count_turns	Gemma-2-9B	153	54.9 (84)	22.6	24.6	75.0	62.3
nav5	adv_count_turns	Llama-3.1-8B	153	85.6 (131)	33.6	9.1	57.3	63.6

model	s1	s2	s3
Gemma-2-9B	97.8 → 92.4 (-5.4) · states 0%	75.5 → 64.5 (-11.0) · states 0%	36.0 → 35.0 (-1.0) · states 0%
Llama-3.1-8B	91.3 → 76.1 (-15.2) · states 5%	56.0 → 55.5 (-0.5) · states 6%	18.5 → 22.5 (+4.0) · states 2%
Mistral-Small-22B	95.7 → 84.8 (-10.9) · states 9%	70.5 → 71.5 (+1.0) · states 20%	37.5 → 35.0 (-2.5) · states 4%
Qwen-2.5-14B	100.0 → 95.7 (-4.3) · states 0%	81.0 → 91.5 (+10.5) · states 0%	67.5 → 59.5 (-8.0) · states 0%
Qwen3-14B OFF	98.9 → 97.8 (-1.1) · states 0%	91.5 → 97.0 (+5.5) · states 0%	73.0 → 71.0 (-2.0) · states 2%
Qwen3-14B ON	100.0 → 100.0 (+0.0) · states 0%	97.0 → 99.5 (+2.5) · states 2%	97.0 → 95.5 (-1.5) · states 2%

Table 98: Date `adv_ignore_shift`: as Table 97.

tier	rule	model	n	states % (n)	acc. if states	acc. if not	rule's answer if states	rule's answer if not
nav5	adv_count_turns	Mistral-Small-22B	153	62.7 (96)	19.8	33.3	80.2	66.7
nav5	adv_count_turns	Qwen-2.5-14B	153	29.4 (45)	71.1	83.3	28.9	16.7
nav5	adv_count_turns	Qwen3-14B OFF	153	62.7 (96)	5.2	22.8	94.8	77.2
nav5	adv_count_turns	Qwen3-14B ON	153	64.7 (99)	71.7	87.0	27.3	13.0
nav20	adv_skip	Gemma-2-9B	100	100.0 (100)	36.0	—	54.0	—
nav20	adv_skip	Llama-3.1-8B	100	100.0 (100)	53.0	—	44.0	—
nav20	adv_skip	Mistral-Small-22B	100	83.0 (83)	49.4	76.5	50.6	23.5
nav20	adv_skip	Qwen-2.5-14B	100	71.0 (71)	70.4	58.6	29.6	41.4
nav20	adv_skip	Qwen3-14B OFF	100	92.0 (92)	60.9	87.5	38.0	12.5
nav20	adv_skip	Qwen3-14B ON	100	80.0 (80)	87.5	85.0	11.2	15.0
nav20	adv_last_step	Gemma-2-9B	60	83.3 (50)	14.0	30.0	86.0	40.0
nav20	adv_last_step	Llama-3.1-8B	60	100.0 (60)	5.0	—	95.0	—
nav20	adv_last_step	Mistral-Small-22B	60	96.7 (58)	6.9	0.0	93.1	100.0
nav20	adv_last_step	Qwen-2.5-14B	60	15.0 (9)	0.0	43.1	100.0	56.9
nav20	adv_last_step	Qwen3-14B OFF	60	100.0 (60)	0.0	—	100.0	—
nav20	adv_last_step	Qwen3-14B ON	60	48.3 (29)	79.3	87.1	20.7	12.9
nav20	adv_count_turns	Gemma-2-9B	52	96.2 (50)	46.0	100.0	54.0	0.0
nav20	adv_count_turns	Llama-3.1-8B	52	73.1 (38)	15.8	21.4	84.2	78.6
nav20	adv_count_turns	Mistral-Small-22B	52	98.1 (51)	41.2	100.0	58.8	0.0
nav20	adv_count_turns	Qwen-2.5-14B	52	69.2 (36)	97.2	100.0	2.8	0.0
nav20	adv_count_turns	Qwen3-14B OFF	52	65.4 (34)	2.9	11.1	97.1	88.9
nav20	adv_count_turns	Qwen3-14B ON	52	53.8 (28)	89.3	100.0	10.7	0.0

Table 103: Stating a deceptive strategy vs following it, date.

tier	rule	model	n	states % (n)	acc. if states	acc. if not	rule's answer if states	rule's answer if not
s1	adv_skip	Gemma-2-9B	152	88.8 (135)	43.0	23.5	n/a	n/a
s1	adv_skip	Llama-3.1-8B	152	100.0 (152)	70.4	—	n/a	n/a
s1	adv_skip	Mistral-Small-22B	152	96.7 (147)	78.9	100.0	n/a	n/a
s1	adv_skip	Qwen-2.5-14B	152	15.1 (23)	95.7	97.7	n/a	n/a
s1	adv_skip	Qwen3-14B OFF	152	21.1 (32)	84.4	95.8	n/a	n/a
s1	adv_skip	Qwen3-14B ON	152	7.2 (11)	81.8	99.3	n/a	n/a
s1	adv_ignore_shift	Gemma-2-9B	92	0.0 (0)	—	92.4	n/a	n/a
s1	adv_ignore_shift	Llama-3.1-8B	92	5.4 (5)	20.0	79.3	n/a	n/a
s1	adv_ignore_shift	Mistral-Small-22B	92	8.7 (8)	12.5	91.7	n/a	n/a
s1	adv_ignore_shift	Qwen-2.5-14B	92	0.0 (0)	—	95.7	n/a	n/a
s1	adv_ignore_shift	Qwen3-14B OFF	92	0.0 (0)	—	97.8	n/a	n/a
s1	adv_ignore_shift	Qwen3-14B ON	92	0.0 (0)	—	100.0	n/a	n/a
s1	adv_wrong_direction	Gemma-2-9B	92	0.0 (0)	—	93.5	n/a	n/a
s1	adv_wrong_direction	Llama-3.1-8B	92	0.0 (0)	—	83.7	n/a	n/a
s1	adv_wrong_direction	Mistral-Small-22B	92	2.2 (2)	0.0	91.1	n/a	n/a
s1	adv_wrong_direction	Qwen-2.5-14B	92	0.0 (0)	—	95.7	n/a	n/a
s1	adv_wrong_direction	Qwen3-14B OFF	92	3.3 (3)	100.0	100.0	n/a	n/a
s1	adv_wrong_direction	Qwen3-14B ON	92	3.3 (3)	100.0	98.9	n/a	n/a
s3	adv_skip	Gemma-2-9B	200	99.5 (199)	3.0	0.0	n/a	n/a
s3	adv_skip	Llama-3.1-8B	200	100.0 (200)	10.0	—	n/a	n/a
s3	adv_skip	Mistral-Small-22B	200	99.5 (199)	23.1	0.0	n/a	n/a
s3	adv_skip	Qwen-2.5-14B	200	12.0 (24)	54.2	55.1	n/a	n/a
s3	adv_skip	Qwen3-14B OFF	200	62.0 (124)	62.9	67.1	n/a	n/a
s3	adv_skip	Qwen3-14B ON	200	6.0 (12)	25.0	90.4	n/a	n/a
s3	adv_ignore_shift	Gemma-2-9B	200	0.0 (0)	—	35.0	n/a	n/a
s3	adv_ignore_shift	Llama-3.1-8B	200	1.5 (3)	0.0	22.8	n/a	n/a
s3	adv_ignore_shift	Mistral-Small-22B	200	4.0 (8)	12.5	35.9	n/a	n/a
s3	adv_ignore_shift	Qwen-2.5-14B	200	0.0 (0)	—	59.5	n/a	n/a
s3	adv_ignore_shift	Qwen3-14B OFF	200	2.0 (4)	50.0	71.4	n/a	n/a
s3	adv_ignore_shift	Qwen3-14B ON	200	1.5 (3)	100.0	95.4	n/a	n/a
s3	adv_wrong_direction	Gemma-2-9B	200	1.5 (3)	0.0	27.4	n/a	n/a
s3	adv_wrong_direction	Llama-3.1-8B	200	1.0 (2)	0.0	18.2	n/a	n/a
s3	adv_wrong_direction	Mistral-Small-22B	200	0.0 (0)	—	36.0	n/a	n/a
s3	adv_wrong_direction	Qwen-2.5-14B	200	0.0 (0)	—	58.5	n/a	n/a
s3	adv_wrong_direction	Qwen3-14B OFF	200	1.0 (2)	50.0	67.2	n/a	n/a
s3	adv_wrong_direction	Qwen3-14B ON	200	1.5 (3)	66.7	81.7	n/a	n/a

model	s1	s2	s3
Gemma-2-9B	97.8 → 93.5 (-4.3) · states 0%	75.5 → 66.5 (-9.0) · states 0%	36.0 → 27.0 (-9.0) · states 2%
Llama-3.1-8B	91.3 → 83.7 (-7.6) · states 0%	56.0 → 58.0 (+2.0) · states 0%	18.5 → 18.0 (-0.5) · states 1%
Mistral-Small-22B	95.7 → 89.1 (-6.5) · states 2%	70.5 → 62.0 (-8.5) · states 2%	37.5 → 36.0 (-1.5) · states 0%
Qwen-2.5-14B	100.0 → 95.7 (-4.3) · states 0%	81.0 → 80.5 (-0.5) · states 0%	67.5 → 58.5 (-9.0) · states 0%
Qwen3-14B OFF	98.9 → 100.0 (+1.1) · states 3%	91.5 → 86.5 (-5.0) · states 5%	73.0 → 67.0 (-6.0) · states 1%
Qwen3-14B ON	100.0 → 98.9 (-1.1) · states 3%	97.0 → 87.5 (-9.5) · states 6%	97.0 → 81.5 (-15.5) · states 2%

Table 99: Date `adv_wrong_direction`: as Table 97.

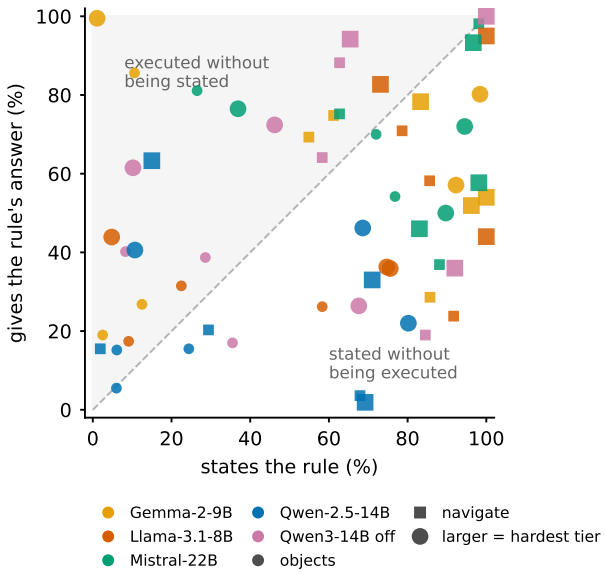


Figure 7: **Saying the rule and doing it are different things.** One dot per model, rule and tier (easiest and hardest): x is the share of answers that restate the rule, y the share that give its answer. Points above the diagonal executed the rule without stating it; points below stated it without executing it. $r = 0.28$ over 60 cells.

Objects, N=11					Navigate, 20 steps			
			gives rule's answer				gives rule's answer	
Rule	Model	states	if stated	if not	Rule	states	if stated	if not
swaps cancel	Gemma-2-9B	1	100	100	forward = backward	100	54	–
	Llama-3.1-8B	5	78	42		100	44	–
	Mistral-22B	37	90	69		83	51	24
	Qwen-2.5-14B	11	45	40		71	30	41
	Qwen3-14B off	10	53	62		92	38	12
last swap only	Gemma-2-9B	98	80	100	last step only	83	86	40
	Llama-3.1-8B	75	36	37		100	95	–
	Mistral-22B	94	75	20		97	93	100
	Qwen-2.5-14B	80	24	14		15	100	57
	Qwen3-14B off	68	35	8		100	100	–
parity	Gemma-2-9B	92	56	67	count turns	96	54	0
	Llama-3.1-8B	76	41	21		73	84	79
	Mistral-22B	90	50	50		98	59	0
	Qwen-2.5-14B	69	59	18		69	3	0
	Qwen3-14B off	46	76	69		65	97	89

Table 100: **Stating a strategy is neither necessary nor sufficient for executing it.** Hardest tier, only instances where the rule's answer is wrong. *states*: % of answers that restate the rule (author-reviewed keyword list, agreement 15/15); *gives rule's answer*: % of answers equal to the rule's output, separately for answers that state it and answers that do not.

G MECHANISM: ADDITIONAL DETAILS AND TABLES

This appendix completes Section 7. It gives the construction of the patching pairs (Section G.1) and the head-level results (Section G.2, Section G.3), how the residual stream is read and where (Section G.4), and every per-size and per-layer probe table behind the numbers the body quotes: at the last prompt token (Section G.5), at the answering position on a corrupted solution (Section G.6) and under a deceptive strategy (Section G.7), the faint trace of the correct item (Section G.8), the depth at which the correct item overtakes the planted one on the rows where the model answers correctly (Section G.9), and the same probes with thinking on (Section G.10). Patching is on Gemma-2-9B (42 layers) and Qwen-2.5-14B (48 layers); probes are on those two, Llama-3.1-8B, and Qwen3-14B in both modes. Depth is always a fraction of the model's layer count, so that models can be overlaid.

G.1 PATCHING: PAIRS, MARGIN AND RESTORATION

A causal question needs two prompts that differ in exactly one thing. The *clean* prompt is the question, the full worked solution ending in its final-state block, and the cue "Therefore, X has the", where X is the queried person, so that the next token is the answer item. The *corrupt* prompt is byte-identical except that X's item on the final-state line is replaced by another option. We keep a pair only if the two prompts tokenize to sequences of the same length that differ at exactly that position; this leaves about 134 eligible problems per size, of which we take 80. At the cue we measure the margin between the logit of the correct item's first token and that of the planted one, positive when the model answers with the clean state and negative when it answers with the corrupted one, and we keep the pairs on which the corruption flipped the model, 72–80 of 80 at every size for both models; on those pairs the clean margin is $+7.0$ to $+10.5$ and the corrupt margin -5.0 to -9.3, so the flips are not marginal. We then run the clean prompt once and cache the residual stream at every layer and position, run the corrupt prompt while overwriting the residual stream at one layer and one position with its cached clean value, and recompute the margin. *Restoration* is the fraction of the clean–corrupt gap this recovers, $(m_{\text{patched}} - m_{\text{corr}})/(m_{\text{clean}} - m_{\text{corr}})$: 1 when the patch restores the clean answer completely, 0 when it does nothing. We sweep every layer at three positions: the swapped state token, the cue token, and a control token half-way between the start of the prompt and the swap, which mentions nothing relevant. Table 10 in the body gives the sweep at $N{=}7$; the other sizes give the same picture, with the two curves crossing at layers 24–26 of 42 in Gemma-2 and 31–32 of 48 in Qwen-2.5, that is, at 0.58–0.66 of depth in both models, and the crossing does not move with N. Two remarks. The pairs are the instances on which the model copied, so the sweep says how the copy happens and not that checking never does; that rests on the behavioural rates of Section 5. And the depth is the same in two unrelated families, which suggests a retrieval stage that belongs to the task rather than to a model.

G.2 PATCHING: THE HAND-OFF IS CARRIED BY A POPULATION OF HEADS

With the pairs of Section G.1 in hand, we ask whether a small set of attention heads implements the transport. We patched individual heads in the band around the crossover (0.45–0.85 of depth; layers 18–34 in Gemma-2 and 21–39 in Qwen-2.5), overwriting each head's contribution to the residual stream at the cue token with its clean value while running the corrupt prompt, and measured restoration as before. No single head does much. The best head in either model restores 0.17–0.24 of the gap, against 1.00 for the full residual at the same position, and the top heads are the same across sizes and sit inside the crossover band (Gemma-2: layer 28 head 12, layer 25 head 7, layer 27 head 15, layer 34 head 15; Qwen-2.5: layer 39 head 23, layer 36 heads 10 and 36, layer 38 head 35). Patching the top-K heads together, ranked by their individual effect, restores the gap roughly linearly in K with no knee: about 0.65 at $K = 8$, each additional head adding 0.05–0.13 without the contribution decaying (Table 104). Attention from the cue token to the state token is likewise spread thin. Only 16 of 272 heads in Gemma-2 and 49 of 760 in Qwen-2.5 attend to it with weight above 0.1, a single Gemma-2 head exceeds 0.3, and the correlation between a head's attention to the state token and its restoration is 0.36 in both models: heads that look at the stated item tend to carry it, but many carry a little.

Table 104: Head-level patching in the crossover band at the cue token, objects $N \in \{5, 7, 11\}$. Restoration from the best single head and from the top-K heads patched jointly, against the full residual patch at the same position; number of heads whose attention from the cue to the state token exceeds a threshold; and the correlation across heads between that attention and single-head restoration. Ranges are over the three sizes.

	Gemma-2-9B	Qwen-2.5-14B
Heads in band (layers)	18–34	21–39
Full residual patch at cue	1.00	1.00
Best single head	0.17–0.24	0.19–0.20
Top-8 heads, joint	≈ 0.65	≈ 0.64
Marginal gain per added head	0.05–0.13	0.05–0.13
Heads with attention cue→state > 0.1	16 / 272	49 / 760
Heads with attention > 0.3	1	0
corr(attention, restoration)	0.34–0.36	0.35–0.36

G.3 Ablation: no head, set of heads or attention edge is necessary

The patching above measures sufficiency. Here we measure necessity, on the clean prompts of the same pairs. Two ablations: *mean* replaces a head's output at the cue token by its mean over the clean prompts of that size; *edge* zeroes the head's attention from the cue token to the swapped state token and renormalises the remaining weights, applied as the exact identity $h' = (h - w_{\text{state}} v_{\text{state}})/(1 - w_{\text{state}})$. *Damage* is $(m_{\text{clean}} - m_{\text{ablated}})/(m_{\text{clean}} - m_{\text{corrupt}})$, and the *flip rate* is the share of pairs on which the stated item no longer outscores the planted one; every pair enters with the stated item ahead, so the unablated flip rate is 0 (the raw argmax at the cue is usually a formatting token, so an argmax-based flip rate is uninformative and is not used). Heads are ranked either by their own damage or by their restoration from the patching sweep; the random baseline draws K heads from the band, 20 draws per K. The bare-prompt specificity control could not be read: at the cue, with no solution in the prompt, the models score near chance (0.12–0.25 against 0.09–0.20), so there is nothing for an ablation to damage. Table 105 summarises and Table 106 gives the top-K curves.

Model	N	method	best head	damage	K=8 damage	smallest K, flip≥0.5	full band, cue	full band, all positions	bare acc. (unablated)
Gemma-2-9B	5	mean	L27.H15	0.080	0.418	none	0.388	0.812	0.237 (0.250)
Gemma-2-9B	7	mean	L27.H15	0.073	0.408	none	0.240	0.840	0.213 (0.120)
Gemma-2-9B	11	mean	L27.H15	0.069	0.378	none	0.189	0.919	0.122 (0.135)
Gemma-2-9B	5	edge	L32.H7	0.013	0.058	none	0.025	0.037	—
Gemma-2-9B	7	edge	L34.H15	0.013	0.052	none	0.000	0.013	—
Gemma-2-9B	11	edge	L34.H15	0.015	0.049	none	0.027	0.054	—
Qwen-2.5-14B	5	mean	L39.H23	0.099	0.387	32	0.575	not run	0.192 (0.247)
Qwen-2.5-14B	7	mean	L39.H23	0.088	0.316	32	0.554	not run	0.189 (0.122)
Qwen-2.5-14B	11	mean	L39.H23	0.076	0.254	none	0.432	0.649	0.095 (0.149)
Qwen-2.5-14B	5	edge	L36.H36	0.024	0.063	none	0.000	0.000	—
Qwen-2.5-14B	7	edge	L36.H36	0.024	0.063	none	0.000	0.000	—
Qwen-2.5-14B	11	edge	L36.H36	0.015	0.038	none	0.000	0.014	—

Table 105: Ablation in the crossover band, clean prompts of the patching pairs. *damage* is for the best single head (its flip rate is 0 in every cell); *full band* columns are flip rates when every head in the band is ablated at the cue only, or at every position. Bare accuracy is an argmax over the option items at the cue with no solution in the prompt, under full-band ablation (unablated in parentheses); chance is $1/N$.

G.4 How the residual stream is read

Positions. We record the residual stream at every layer at three positions of the prompt and one of the generation. The first is the last prompt token, from which the model produces its first output token. The second is the line of the provided state block that mentions the person the question asks about, where the relevant item is literally written; it serves as a check that the item is readable at all. The third is the cue position of the patching experiment, "Therefore, X has the". The fourth is the position at which the model begins to write its answer.

Model	method	K=1	K=2	K=4	K=8	K=16	K=32
Gemma-2-9B	mean, top-K by damage	0.07–0.08	0.12–0.15	0.23–0.27	0.38–0.42	0.49–0.53	0.50–0.58
Gemma-2-9B	mean, top-K by restoration	0.04–0.06	0.09–0.15	0.23–0.27	0.34–0.42	0.45–0.53	0.55–0.59
Gemma-2-9B	mean, random K (mean)	0.00	0.00	0.00–0.01	0.01–0.02	0.03	0.02
Gemma-2-9B	edge, top-K by damage	0.01	0.02–0.03	0.03–0.04	0.05–0.06	0.06–0.09	0.07–0.10
Qwen-2.5-14B	mean, top-K by damage	0.08–0.10	0.11–0.14	0.20–0.25	0.25–0.39	0.38–0.48	0.50–0.63
Qwen-2.5-14B	mean, top-K by restoration	0.08–0.10	0.10–0.14	0.18–0.24	0.29–0.38	0.41–0.53	0.49–0.60
Qwen-2.5-14B	mean, random K (mean)	0.00	0.00	0.00	0.00	0.01	0.01
Qwen-2.5-14B	edge, top-K by damage	0.02	0.02–0.03	0.03–0.05	0.04–0.06	0.04–0.08	0.06–0.10

Table 106: Damage from ablating the top-K heads jointly, ranges over $N \in \{5, 7, 11\}$. The random-K row is the mean over 20 draws; its 95th percentile stays below 0.18 at every K. Ranking by damage and by restoration give the same curve, and neither has a knee.

Locating the answering position. We replay each saved generation through the model: the reconstructed prompt and the generation's own tokens are passed in a single forward pass with no generation step, and the residual stream is taken at the index preceding the first mention of any answer content, an item name or an option letter, whichever comes first. The reading therefore does not depend on how the generation was produced, and sampled and greedily decoded models are read on the same footing. Two checks say that the reading is taken where we intend. First, on 99.0% of the 22,982 replayed rows (five models, five sizes, eleven conditions) the token at the anchor is the first token of the answer content; on the remaining 1% a single token spans the markdown just before the answer and the reading is taken one token earlier, still before any answer content, so the error can only make less of the answer readable. Second, the model's argmax at the anchor equals the saved next token on 98.2% of rows; of the 582 mismatches, 445 name the same item written differently (a bold span, a capital, an inflection), and the 137 that put a different word there are 0.24% of the rows in the greedily decoded cells, where they are what bfloat16 batching gives at a near tie, and the rest are in the sampled Qwen3 cells, where a divergence is expected. No row is excluded on either check; the only rows excluded are generations with no extractable answer, and hence no anchor: 202 of Gemma-2's 7,998 rows, 84 of Llama's 4,316, none of Qwen-2.5's 11,000, and 4 of Qwen3's 8,796. Table 107 gives the per-cell agreement for the greedily decoded models and Table 108 the rows kept in the sampled Qwen3 cells, with the split between rows that open their answer with an item name and with an option letter.

Probes. Probes are multinomial logistic regressions on the residual stream at a given layer, predicting the identity of an item. They are fit within each surface domain, since the three domains have disjoint item vocabularies, and every prediction is cross-fitted, so that no problem is scored by a probe that saw it. Chance is the majority-class rate of the test labels and is printed with every readout. Every number in the body is the accuracy at the best layer; the tables below give that depth in parentheses, and the per-layer profiles where they matter. The *deposited-item* probe is trained on prompts with a correct state block to read the item stated for the queried person. The *own-answer* probe is trained on prompts with no trace at all, restricted to instances the model solved, to read the answer the model computes for itself; training it on all no-trace rows instead is the control against reading the answer off the question text.

Table 107: Agreement between the replayed model and the saved generation at the answering position, greedily decoded models. "strict" is the share of rows whose argmax at the anchor equals the saved next token; "misses" counts the rest; "genuine" counts the misses that put a different word at the anchor rather than the same item written differently; "whole prefix" is the agreement rate over all positions before the answer. The statistic is a check on prompt reconstruction, and no row is excluded on it; the probe readings are taken by teacher-forcing at the position the saved generation occupies (Appendix G.4). The sampled Qwen3 cells are in Table 108.

Model	Size	Condition	strict	misses	genuine	whole prefix
Gemma-2-9B	N=11	corrupted expert	0.990	2	0	1.000
Gemma-2-9B	N=11	expert (state)	1.000	0	0	1.000
Gemma-2-9B	N=11	no CoT	1.000	0	0	1.000
Gemma-2-9B	N=3	corrupted expert	0.995	1	1	0.995
Gemma-2-9B	N=3	expert (state)	1.000	0	0	1.000
Gemma-2-9B	N=3	no CoT	1.000	0	0	1.000
Gemma-2-9B	N=7	corrupted expert	0.995	1	1	0.995
Gemma-2-9B	N=7	expert (state)	1.000	0	0	1.000
Gemma-2-9B	N=7	no CoT	0.995	1	0	1.000

Model	Size	Condition	strict	misses	genuine	whole prefix
Llama-3.1-8B	N=11	corrupted expert	0.985	3	1	0.995
Llama-3.1-8B	N=7	corrupted expert	0.995	1	1	0.995
Llama-3.1-8B	N=7	no CoT	0.995	1	0	1.000
Qwen-2.5-14B	N=11	corrupted expert	0.985	3	0	1.000
Qwen-2.5-14B	N=11	expert (state)	0.995	1	0	1.000
Qwen-2.5-14B	N=11	no CoT	0.980	4	0	1.000
Qwen-2.5-14B	N=3	corrupted expert	0.990	2	0	1.000
Qwen-2.5-14B	N=3	expert (state)	0.990	2	0	1.000
Qwen-2.5-14B	N=3	no CoT	0.995	1	0	1.000
Qwen-2.5-14B	N=5	corrupted expert	0.995	1	1	0.995
Qwen-2.5-14B	N=5	expert (state)	0.965	7	2	0.990
Qwen-2.5-14B	N=5	no CoT	0.975	5	1	0.995
Qwen-2.5-14B	N=7	corrupted expert	0.995	1	0	1.000
Qwen-2.5-14B	N=7	expert (state)	0.990	2	0	1.000
Qwen-2.5-14B	N=7	no CoT	0.970	6	1	0.995
Qwen-2.5-14B	N=9	corrupted expert	1.000	0	0	1.000
Qwen-2.5-14B	N=9	expert (state)	1.000	0	0	1.000
Qwen-2.5-14B	N=9	no CoT	0.975	5	2	0.990
Qwen3-14B (off)	N=11	corrupted expert	0.990	2	2	0.990
Qwen3-14B (off)	N=11	expert (state)	0.990	2	0	1.000
Qwen3-14B (off)	N=11	no CoT	0.935	13	1	0.995
Qwen3-14B (off)	N=7	corrupted expert	0.995	1	0	1.000
Qwen3-14B (off)	N=7	expert (state)	0.995	1	1	0.995
Qwen3-14B (off)	N=7	no CoT	0.915	17	5	0.975
Qwen3-14B (on)	N=11	expert (state)	0.990	2	2	0.990
Qwen3-14B (on)	N=11	no CoT	0.930	14	10	0.950
Qwen3-14B (on)	N=7	corrupted expert	0.860	28	2	0.990
Qwen3-14B (on)	N=7	expert (state)	1.000	0	0	1.000
Qwen3-14B (on)	N=7	no CoT	0.920	16	10	0.950

Table 108: Rows read at the answering position for the two sampled Qwen3 modes, per cell: rows kept and dropped (a row is dropped only when its generation has no extractable answer), the share of rows whose argmax at the anchor equals the saved next token (reported, never used as a condition, since the saved token is a sample), and how many rows open their answer with an item name and how many with an option letter. Over the 22 cells of each mode, 4,399 (off) and 4,397 (on) rows are kept, and agreement ranges from 0.825 to 1.000 with medians 0.965 and 0.972.

Mode	N	Condition	kept	dropped	agreement	item-first	letter-first
off	11	adv_skip	200	0	0.945	155	45
off	11	correct_strategy	200	0	0.970	142	58
off	11	corrupted_expert	200	0	0.990	187	13
off	11	expert_stripped	200	0	0.990	192	8
off	11	no_cot	200	0	0.935	171	29
off	3	adv_skip	200	0	0.940	194	6
off	3	corrupted_expert	200	0	0.975	188	12
off	3	expert_stripped	200	0	0.960	180	20
off	3	no_cot	200	0	0.975	136	64
off	5	adv_skip	199	1	0.859	174	25
off	5	corrupted_expert	200	0	0.990	175	25
off	5	expert_stripped	200	0	1.000	196	4
off	5	no_cot	200	0	0.885	173	27
off	7	adv_skip	200	0	0.825	150	50
off	7	correct_strategy	200	0	0.920	163	37
off	7	corrupted_expert	200	0	0.995	182	18
off	7	expert_stripped	200	0	0.995	185	15
off	7	no_cot	200	0	0.915	177	23
off	9	adv_skip	200	0	0.900	154	46
off	9	corrupted_expert	200	0	1.000	189	11
off	9	expert_stripped	200	0	0.985	181	19
off	9	no_cot	200	0	0.915	172	28
on	11	adv_skip	200	0	0.975	174	26
on	11	correct_strategy	199	1	0.955	181	18
on	11	corrupted_expert	199	1	0.879	195	4
on	11	expert_stripped	200	0	0.990	198	2
on	11	no_cot	200	0	0.930	146	54
on	3	adv_skip	200	0	0.975	192	8
on	3	corrupted_expert	200	0	0.975	197	3
on	3	expert_stripped	200	0	1.000	200	0
on	3	no_cot	200	0	0.985	187	13
on	5	adv_skip	200	0	0.975	191	9
on	5	corrupted_expert	200	0	0.970	200	0
on	5	expert_stripped	200	0	1.000	199	1
on	5	no_cot	200	0	0.960	170	30
on	7	adv_skip	199	1	0.945	173	26
on	7	correct_strategy	200	0	0.995	191	9

Mode	N	Condition	kept	dropped	agreement	item-first	letter-first
on	7	corrupted_expert	200	0	0.860	196	4
on	7	expert_stripped	200	0	1.000	199	1
on	7	no_cot	200	0	0.920	162	38
on	9	adv_skip	200	0	0.955	165	35
on	9	corrupted_expert	200	0	0.870	198	2
on	9	expert_stripped	200	0	1.000	198	2
on	9	no_cot	200	0	0.905	170	30

G.5 The last prompt token

Table 109 gives what is readable at the last prompt token: the item on the last line of the state block and the item of the queried person, each against its chance. Table 110 shows that, within a size, readability of the queried item does not depend on how many lines back its line sits; the apparent decay with N in pooled numbers is the change in the number of classes. Table 111 examines the one size at which the queried item is above chance, $N = 3$, and finds that the signal survives neither corruption nor the own-answer probe. Table 112 shows the item is readable at the queried person's own line, at shallow depth, on clean and corrupted blocks alike.

Table 109: What is readable at the last prompt token (last prompt token): the item on the last line of the provided state block, and the item of the queried person, each against its majority-class chance.

Model	Size	last line	chance	queried person	chance
Gemma-2-9B	N=11	0.985	0.209	0.130	0.149
Gemma-2-9B	N=3	1.000	0.403	0.840	0.418
Gemma-2-9B	N=7	1.000	0.164	0.180	0.179
Llama-3.1-8B	N=11	0.970	0.209	0.130	0.149
Llama-3.1-8B	N=7	1.000	0.164	0.190	0.179
Qwen-2.5-14B	N=11	0.995	0.209	0.130	0.149
Qwen-2.5-14B	N=3	1.000	0.403	0.630	0.418
Qwen-2.5-14B	N=5	1.000	0.284	0.295	0.269
Qwen-2.5-14B	N=7	1.000	0.164	0.170	0.179
Qwen-2.5-14B	N=9	1.000	0.164	0.215	0.179
Qwen3-14B (off)	N=11	0.995	0.209	0.130	0.149
Qwen3-14B (off)	N=7	1.000	0.164	0.205	0.179
Qwen3-14B (on)	N=11	0.995	0.209	0.130	0.149
Qwen3-14B (on)	N=7	1.000	0.164	0.215	0.179

Table 110: Queried-person decodability at the last prompt token by the number of lines between the queried person's line and the end of the state block. Within a size the effect is flat; the pooled decay is the change in the number of classes.

Model	Size	0 lines	1–2	3–5	6+
Gemma-2-9B	N=11	0.167	0.081	0.160	0.129
Gemma-2-9B	N=3	0.908	0.874	–	–
Gemma-2-9B	N=7	0.172	0.127	0.240	0.152
Qwen-2.5-14B	N=11	0.167	0.081	0.160	0.129
Qwen-2.5-14B	N=3	0.692	0.622	–	–
Qwen-2.5-14B	N=5	0.364	0.200	0.368	–
Qwen-2.5-14B	N=7	0.034	0.206	0.227	0.091
Qwen-2.5-14B	N=9	0.125	0.179	0.257	0.224

G.6 The answering position on a corrupted solution

Section 7 reports that at the answering position the planted item is readable at 0.92 or better, that it becomes readable at the depth where patching found the hand-off, and that it is still readable on the rows where the model answered correctly. Table 113 pools the deposited-item probe over sizes and Table 114 gives it per size. Table 115 splits the rows by whether the answer opened with the item name or with an option letter: on item-first rows the item is readable before it is written. Table 116

Table 111: Is the above-chance signal at $N = 3$ the deposit or the model's own computation? Probes at the last prompt token. Neither: the deposited-item probe reads neither the planted nor the correct item on corrupted prompts, and the own-answer probe fails on its own held-out prompts.

Model	Probe	Test	depth	acc [95% CI]	chance
Gemma-2-9B	deposited	corrupted/implied	0.95	0.455 [0.39, 0.52]	0.403
Gemma-2-9B	deposited	corrupted/gold	0.19	0.330 [0.27, 0.40]	0.418
Gemma-2-9B	deposited	expert_stripped/deposited (held out)	0.67	0.840 [0.78, 0.88]	0.418
Gemma-2-9B	gold[solved]	expert_stripped/gold	0.95	0.375 [0.31, 0.44]	0.418
Gemma-2-9B	gold[solved]	corrupted/gold	0.95	0.375 [0.31, 0.44]	0.418
Gemma-2-9B	gold[solved]	no_cot/gold (held out)	0.10	0.400 [0.34, 0.47]	0.418
Qwen-2.5-14B	deposited	corrupted/implied	0.67	0.415 [0.35, 0.48]	0.403
Qwen-2.5-14B	deposited	corrupted/gold	0.50	0.380 [0.32, 0.45]	0.418
Qwen-2.5-14B	deposited	expert_stripped/deposited (held out)	0.67	0.630 [0.56, 0.69]	0.418
Qwen-2.5-14B	gold[solved]	expert_stripped/gold	0.67	0.400 [0.34, 0.47]	0.418
Qwen-2.5-14B	gold[solved]	corrupted/gold	0.67	0.405 [0.34, 0.47]	0.418
Qwen-2.5-14B	gold[solved]	no_cot/gold (held out)	0.25	0.355 [0.29, 0.42]	0.418

Table 112: The item is readable at the queried person's own line at early depth, on clean and corrupted blocks alike.

Model	Size	Test	depth	acc [95% CI]	chance
Qwen-2.5-14B	N=11	entity_line/deposited (held out)	0.08	0.930 [0.89, 0.96]	0.149
Qwen-2.5-14B	N=11	corrupted_entity_line/implied	0.25	1.000 [0.98, 1.00]	0.134
Qwen-2.5-14B	N=3	entity_line/deposited (held out)	0.08	1.000 [0.98, 1.00]	0.418
Qwen-2.5-14B	N=3	corrupted_entity_line/implied	0.08	1.000 [0.98, 1.00]	0.403
Qwen-2.5-14B	N=7	entity_line/deposited (held out)	0.25	0.930 [0.89, 0.96]	0.179
Qwen-2.5-14B	N=7	corrupted_entity_line/implied	0.08	1.000 [0.98, 1.00]	0.164
Gemma-2-9B	N=11	entity_line/deposited (held out)	0.10	0.935 [0.89, 0.96]	0.149
Gemma-2-9B	N=11	corrupted_entity_line/implied	0.10	1.000 [0.98, 1.00]	0.134
Gemma-2-9B	N=3	entity_line/deposited (held out)	0.10	1.000 [0.98, 1.00]	0.418
Gemma-2-9B	N=3	corrupted_entity_line/implied	0.10	1.000 [0.98, 1.00]	0.403
Gemma-2-9B	N=7	entity_line/deposited (held out)	0.29	0.930 [0.89, 0.96]	0.179
Gemma-2-9B	N=7	corrupted_entity_line/implied	0.10	1.000 [0.98, 1.00]	0.164

gives the depth at which the planted item becomes readable at the cue position, against the 0.58–0.66 band in which patching found the hand-off. Table 117 is the own-answer probe, a probe trained on prompts with no solution to read the answer the model computes for itself: on corrupted prompts it finds the planted item and not the correct one. Table 118 reads both items inside each behavioural outcome bin; the "gold" bin, where the model answered correctly despite the corruption, is the resisted-rows result of Section 7, and it holds 17 and 11 rows for Gemma-2 and Qwen-2.5, so the reading is coarse.

Table 113: Deposited-item probe at the answering position, pooled over sizes. The probe is trained on prompts with a correct state block to read the item stated for the queried person, and tested on held-out prompts of the same kind, on corrupted prompts scored against the planted item and against the correct item, on prompts with a wrong strategy in front of a correct solution, and on prompts with nothing added. Entries are accuracy at the best depth (depth as a fraction of layers in parentheses); chance is the majority-class rate.

Model	held-out deposited	corrupted: planted	corrupted: correct	wrong strategy + solution: correct	no CoT: correct	chance	n
Qwen-2.5-14B	0.966 (0.81)	0.965 (0.81)	0.188 (0.44)	0.987 (0.81)	0.246 (1.00)	0.21	1000
Gemma-2-9B	0.995 (0.86)	0.928 (0.86)	0.192 (0.00)	0.873 (0.79)	0.347 (0.93)	0.23	600
Llama-3.1-8B	0.789 (1.00)	0.925 (0.94)	0.102 (0.28)	–	0.133 (0.84)	0.17	400
Qwen3-14B (off)	0.635 (0.97)	0.648 (1.00)	0.140 (0.60)	–	0.582 (0.82)	0.17	400
Qwen3-14B (on)	0.955 (0.75)	0.697 (0.97)	0.120 (1.00)	–	0.438 (1.00)	0.17	399

Table 114: Deposited-item probe at the answering position, per size.

Model	Size	Test	depth	acc [95% CI]	n
Gemma-2-9B	N=11	corrupted/gold	0.50	0.152 [0.08, 0.26]	66
Gemma-2-9B	N=3	corrupted/gold	0.00	0.373 [0.27, 0.49]	67
Gemma-2-9B	N=7	corrupted/gold	0.21	0.182 [0.11, 0.29]	66
Gemma-2-9B	N=11	corrupted/implied	0.86	0.970 [0.90, 0.99]	67
Gemma-2-9B	N=3	corrupted/implied	0.64	0.955 [0.88, 0.98]	66

Model	Size	Test	depth	acc [95% CI]	n
Gemma-2-9B	N=7	corrupted/implied	0.71	0.985 [0.92, 1.00]	67
Gemma-2-9B	N=11	expert_stripped/deposited (held out)	0.79	1.000 [0.95, 1.00]	67
Gemma-2-9B	N=3	expert_stripped/deposited (held out)	0.62	1.000 [0.95, 1.00]	67
Gemma-2-9B	N=7	expert_stripped/deposited (held out)	0.64	1.000 [0.95, 1.00]	67
Gemma-2-9B	N=11	no_cot/gold	0.71	0.418 [0.31, 0.54]	67
Gemma-2-9B	N=3	no_cot/gold	0.14	0.379 [0.27, 0.50]	66
Gemma-2-9B	N=7	no_cot/gold	0.71	0.672 [0.55, 0.77]	67
Gemma-2-9B	N=11	wrong_strategy_correct/gold	0.79	1.000 [0.95, 1.00]	67
Gemma-2-9B	N=3	wrong_strategy_correct/gold	0.57	0.910 [0.82, 0.96]	67
Gemma-2-9B	N=7	wrong_strategy_correct/gold	0.71	0.985 [0.92, 1.00]	67
Llama-3.1-8B	N=11	corrupted/gold	0.47	0.134 [0.07, 0.24]	67
Llama-3.1-8B	N=7	corrupted/gold	0.00	0.149 [0.08, 0.25]	67
Llama-3.1-8B	N=11	corrupted/implied	0.94	0.970 [0.90, 0.99]	67
Llama-3.1-8B	N=7	corrupted/implied	0.75	0.985 [0.92, 1.00]	67
Llama-3.1-8B	N=11	expert_stripped/deposited (held out)	1.00	0.821 [0.71, 0.89]	67
Llama-3.1-8B	N=7	expert_stripped/deposited (held out)	0.62	0.925 [0.84, 0.97]	67
Llama-3.1-8B	N=11	no_cot/gold	0.09	0.136 [0.07, 0.24]	66
Llama-3.1-8B	N=7	no_cot/gold	0.84	0.224 [0.14, 0.34]	67
Qwen-2.5-14B	N=11	corrupted/gold	0.12	0.149 [0.08, 0.25]	67
Qwen-2.5-14B	N=3	corrupted/gold	0.00	0.418 [0.31, 0.54]	67
Qwen-2.5-14B	N=5	corrupted/gold	0.06	0.299 [0.20, 0.42]	67
Qwen-2.5-14B	N=7	corrupted/gold	0.50	0.242 [0.15, 0.36]	66
Qwen-2.5-14B	N=9	corrupted/gold	0.50	0.179 [0.11, 0.29]	67
Qwen-2.5-14B	N=11	corrupted/implied	0.94	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=3	corrupted/implied	0.69	0.970 [0.90, 0.99]	66
Qwen-2.5-14B	N=5	corrupted/implied	0.88	1.000 [0.94, 1.00]	66
Qwen-2.5-14B	N=7	corrupted/implied	0.75	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=9	corrupted/implied	0.81	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=11	expert_stripped/deposited (held out)	0.81	0.970 [0.90, 0.99]	67
Qwen-2.5-14B	N=3	expert_stripped/deposited (held out)	0.62	1.000 [0.94, 1.00]	66
Qwen-2.5-14B	N=5	expert_stripped/deposited (held out)	0.69	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=7	expert_stripped/deposited (held out)	0.75	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=9	expert_stripped/deposited (held out)	0.75	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=11	no_cot/gold	1.00	0.433 [0.32, 0.55]	67
Qwen-2.5-14B	N=3	no_cot/gold	0.56	0.394 [0.28, 0.52]	66
Qwen-2.5-14B	N=5	no_cot/gold	1.00	0.343 [0.24, 0.46]	67
Qwen-2.5-14B	N=7	no_cot/gold	0.69	0.433 [0.32, 0.55]	67
Qwen-2.5-14B	N=9	no_cot/gold	1.00	0.433 [0.32, 0.55]	67
Qwen-2.5-14B	N=11	wrong_strategy_correct/gold	0.81	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=3	wrong_strategy_correct/gold	0.69	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=5	wrong_strategy_correct/gold	0.69	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=7	wrong_strategy_correct/gold	0.75	1.000 [0.95, 1.00]	67
Qwen-2.5-14B	N=9	wrong_strategy_correct/gold	0.75	1.000 [0.95, 1.00]	67
Qwen3-14B (off)	N=11	corrupted/gold	0.00	0.182 [0.11, 0.29]	66
Qwen3-14B (off)	N=7	corrupted/gold	0.00	0.212 [0.13, 0.33]	66
Qwen3-14B (off)	N=11	corrupted/implied	1.00	0.925 [0.84, 0.97]	67
Qwen3-14B (off)	N=7	corrupted/implied	0.75	0.687 [0.57, 0.79]	67
Qwen3-14B (off)	N=11	expert_stripped/deposited (held out)	0.90	0.910 [0.82, 0.96]	67
Qwen3-14B (off)	N=7	expert_stripped/deposited (held out)	0.75	0.851 [0.75, 0.92]	67
Qwen3-14B (off)	N=11	no_cot/gold	0.90	0.791 [0.68, 0.87]	67
Qwen3-14B (off)	N=7	no_cot/gold	0.75	0.925 [0.84, 0.97]	67
Qwen3-14B (on)	N=11	corrupted/gold	0.15	0.152 [0.08, 0.26]	66
Qwen3-14B (on)	N=7	corrupted/gold	0.45	0.194 [0.12, 0.30]	67
Qwen3-14B (on)	N=11	corrupted/implied	0.82	0.761 [0.65, 0.85]	67
Qwen3-14B (on)	N=7	corrupted/implied	0.75	0.716 [0.60, 0.81]	67
Qwen3-14B (on)	N=11	expert_stripped/deposited (held out)	0.75	1.000 [0.95, 1.00]	67
Qwen3-14B (on)	N=7	expert_stripped/deposited (held out)	0.75	1.000 [0.95, 1.00]	67
Qwen3-14B (on)	N=11	no_cot/gold	0.97	0.910 [0.82, 0.96]	67
Qwen3-14B (on)	N=7	no_cot/gold	0.82	0.627 [0.51, 0.73]	67

Table 115: Deposited-item probe at the answering position split by whether the row's answer content opened with the item name or with an option letter. Accuracy on item-first rows shows the item is readable before it is written.

Model	Size	Test	first content	depth	acc	n
Gemma-2-9B	N=11	corrupted/implied	item	0.62	1.000	1
Gemma-2-9B	N=11	corrupted/implied	letter	0.86	0.953	43
Gemma-2-9B	N=11	expert_stripped/deposited (held out)	item	0.62	1.000	2
Gemma-2-9B	N=11	expert_stripped/deposited (held out)	letter	0.71	1.000	44
Gemma-2-9B	N=3	corrupted/implied	item	0.62	0.625	16
Gemma-2-9B	N=3	corrupted/implied	letter	0.71	1.000	48
Gemma-2-9B	N=3	expert_stripped/deposited (held out)	item	0.62	1.000	1
Gemma-2-9B	N=3	expert_stripped/deposited (held out)	letter	0.62	1.000	67
Gemma-2-9B	N=7	corrupted/implied	item	0.71	1.000	4
Gemma-2-9B	N=7	corrupted/implied	letter	0.71	0.985	66

Model	Size	Test	first content	depth	acc	n
Gemma-2-9B	N=7	expert_stripped/deposited (held out)	item	0.62	1.000	3
Gemma-2-9B	N=7	expert_stripped/deposited (held out)	letter	0.64	1.000	64
Llama-3.1-8B	N=11	corrupted/implied	item	0.94	0.970	67
Llama-3.1-8B	N=11	corrupted/implied	letter	1.00	1.000	3
Llama-3.1-8B	N=11	expert_stripped/deposited (held out)	item	1.00	0.864	59
Llama-3.1-8B	N=11	expert_stripped/deposited (held out)	letter	1.00	0.500	8
Llama-3.1-8B	N=7	corrupted/implied	item	0.75	0.985	67
Llama-3.1-8B	N=7	corrupted/implied	letter	0.84	1.000	6
Llama-3.1-8B	N=7	expert_stripped/deposited (held out)	item	0.62	0.938	64
Llama-3.1-8B	N=7	expert_stripped/deposited (held out)	letter	0.62	0.667	3
Qwen-2.5-14B	N=11	corrupted/implied	item	0.94	1.000	67
Qwen-2.5-14B	N=11	expert_stripped/deposited (held out)	item	0.81	0.970	67
Qwen-2.5-14B	N=3	corrupted/implied	item	0.69	0.970	66
Qwen-2.5-14B	N=3	expert_stripped/deposited (held out)	item	0.62	1.000	66
Qwen-2.5-14B	N=5	corrupted/implied	item	0.88	1.000	66
Qwen-2.5-14B	N=5	expert_stripped/deposited (held out)	item	0.69	1.000	67
Qwen-2.5-14B	N=5	expert_stripped/deposited (held out)	letter	0.62	0.000	1
Qwen-2.5-14B	N=7	corrupted/implied	item	0.75	1.000	67
Qwen-2.5-14B	N=7	expert_stripped/deposited (held out)	item	0.75	1.000	67
Qwen-2.5-14B	N=9	corrupted/implied	item	0.81	1.000	67
Qwen-2.5-14B	N=9	expert_stripped/deposited (held out)	item	0.75	1.000	67
Qwen3-14B (off)	N=11	corrupted/implied	item	1.00	0.925	67
Qwen3-14B (off)	N=11	corrupted/implied	letter	0.75	0.538	13
Qwen3-14B (off)	N=11	expert_stripped/deposited (held out)	item	0.90	0.910	67
Qwen3-14B (off)	N=11	expert_stripped/deposited (held out)	letter	0.75	0.500	8
Qwen3-14B (off)	N=7	corrupted/implied	item	1.00	0.755	49
Qwen3-14B (off)	N=7	corrupted/implied	letter	0.75	0.556	18
Qwen3-14B (off)	N=7	expert_stripped/deposited (held out)	item	0.90	0.846	52
Qwen3-14B (off)	N=7	expert_stripped/deposited (held out)	letter	0.75	0.933	15
Qwen3-14B (on)	N=11	corrupted/implied	item	0.82	0.761	67
Qwen3-14B (on)	N=11	corrupted/implied	letter	0.62	0.500	4
Qwen3-14B (on)	N=11	expert_stripped/deposited (held out)	item	0.75	1.000	67
Qwen3-14B (on)	N=11	expert_stripped/deposited (held out)	letter	0.62	0.000	2
Qwen3-14B (on)	N=7	corrupted/implied	item	0.75	0.714	63
Qwen3-14B (on)	N=7	corrupted/implied	letter	0.62	1.000	4
Qwen3-14B (on)	N=7	expert_stripped/deposited (held out)	item	0.75	1.000	67
Qwen3-14B (on)	N=7	expert_stripped/deposited (held out)	letter	0.62	1.000	1

G.7 The answering position under a deceptive strategy

The strategy is *swaps cancel* (`adv_skip`), whose output is the queried person's initial item. Table 119 reads both probes on prompts carrying the rule, scored against that item and against the correct one; Table 120 reads the same probes with no strategy and with the correct one, which is how we know the rule put the item there (the body's "on a correct strategy this item is essentially at chance"); Table 121 restricts to the rows where the model did not give the rule's answer, the 349 problems of Qwen-2.5 and the two Qwen3 modes on which the body's 0.77 and 0.51–0.58 are read. Gemma-2 resists the rule on seven problems, too few to read. The behavioural control, how often the rule's answer is produced without the rule, is Tables 95 and 96 in Appendix F. This is one rule on one task family; we expect, and do not claim, the same for the other rules.

G.8 Is the correct item present at the answering position?

The deposited-item probe reads the correct item at or below chance on corrupted prompts, but it was never trained to find it. We therefore trained a probe on the corrupted prompts themselves to read the correct item, swept over every layer, at the answering position and at the last prompt token (Table 122). It finds the item, faintly: 0.34–0.36 against a chance of 0.15 for the three instruct models on which it was run, on the rows the model copied as much as on the pooled rows, and only in the last layers (Table 123). It does not reach the output: on copied rows the correct item is the runner-up of the output distribution on 4% of rows and its probability is near zero (Table 124). So the correct value is computed to some degree and never selected; the body does not build on this beyond noting it. Probes are per surface domain and cross-fitted by problem, as everywhere else; *random other* is the same probe trained to read a fixed random item that is neither correct nor planted, and *shuffled* the 95th percentile over 20 permutations of the correct-item labels. Rows are the $N \geq 5$ pool.

Table 116: Cue position (cue position, "Therefore, X has the"): peak decodability, and the depth at which the probe first reaches 90% of its peak and halfway from chance to its peak, against the 0.58–0.66 patching band.

Model	Size	Test	peak	at depth	0.9-emergence	half-emergence	in band	chance
Gemma-2-9B	N=11	cue/deposited (held out)	0.875	0.86	–	0.64	True	0.149
Gemma-2-9B	N=11	cue/implied	0.850	0.93	–	0.71	False	0.134
Gemma-2-9B	N=3	cue/deposited (held out)	1.000	0.57	0.57	0.50	False	0.418
Gemma-2-9B	N=3	cue/implied	0.970	0.64	0.57	0.50	False	0.403
Gemma-2-9B	N=7	cue/deposited (held out)	0.990	0.79	0.64	0.57	False	0.179
Gemma-2-9B	N=7	cue/implied	0.960	0.71	0.64	0.57	False	0.164
Qwen-2.5-14B	N=11	cue/deposited (held out)	0.940	0.81	0.81	0.69	False	0.149
Qwen-2.5-14B	N=11	cue/implied	0.955	0.81	0.81	0.69	False	0.134
Qwen-2.5-14B	N=3	cue/deposited (held out)	1.000	0.75	0.62	0.62	True	0.418
Qwen-2.5-14B	N=3	cue/implied	1.000	0.75	0.62	0.62	True	0.403
Qwen-2.5-14B	N=7	cue/deposited (held out)	1.000	0.81	0.75	0.69	False	0.179
Qwen-2.5-14B	N=7	cue/implied	0.990	0.75	0.75	0.69	False	0.164

Table 117: Own-answer probe at the answering position. The probe is trained on prompts with no trace to read the item the model itself answers (either on the rows it solved, or on all rows as a control against reading the answer off the question text). On corrupted prompts it reads the correct item at or below chance and the planted item well above it. Pooled over sizes; accuracy at the best depth.

Model	Probe trained on	no CoT (held out)	corrupted: correct	corrupted (copied rows): correct	corrupted: planted	expert state: correct	adv. skip: correct	chance
Qwen-2.5-14B	solved rows	0.363	0.146	0.147	0.399	0.416	0.170	0.19
Qwen-2.5-14B	all rows	0.319	0.143	0.135	0.393	0.412	0.136	0.19
Gemma-2-9B	solved rows	0.513	0.132	0.109	0.568	0.653	0.180	0.22
Gemma-2-9B	all rows	0.303	0.142	0.136	0.428	0.540	0.175	0.22
Llama-3.1-8B	solved rows	0.133	0.112	0.118	0.155	0.157	–	0.15
Llama-3.1-8B	all rows	0.118	0.110	0.111	0.205	0.162	–	0.15
Qwen3-14B (off)	solved rows	0.626	0.105	0.107	0.538	0.580	0.138	0.15
Qwen3-14B (off)	all rows	0.620	0.112	0.114	0.532	0.600	0.135	0.15
Qwen3-14B (on)	solved rows	0.428	0.133	0.154	0.499	0.660	0.306	0.15
Qwen3-14B (on)	all rows	0.432	0.133	0.111	0.506	0.675	0.311	0.15

Table 118: Planted-item and correct-item decodability at the answering position inside each behavioural adoption bin (rows scored as in the main experiment; the domain with most rows in the bin is shown). In the "gold" bin the model answered correctly despite the corruption, and the planted item is still readable.

Model	Domain	Bin	depth	n	planted	correct
Gemma-2-9B	gifts	gold	0.71	17	0.824	0.176
Gemma-2-9B	soccer	implied	0.79	152	0.987	0.007
Gemma-2-9B	balls	other	0.79	36	0.972	0.028
Llama-3.1-8B	balls	gold	1.00	3	1.000	0.000
Llama-3.1-8B	gifts	implied	0.94	130	0.992	0.000
Llama-3.1-8B	balls	other	0.75	4	0.750	0.000
Qwen-2.5-14B	gifts	gold	0.94	11	0.818	0.182
Qwen-2.5-14B	balls	implied	0.81	306	0.997	0.000
Qwen-2.5-14B	gifts	other	0.81	18	0.944	0.056
Qwen3-14B (off)	gifts	gold	0.62	2	0.500	0.500
Qwen3-14B (off)	balls	implied	1.00	134	0.806	0.015
Qwen3-14B (off)	gifts	other	0.00	1	1.000	0.000
Qwen3-14B (on)	balls	gold	0.82	102	0.637	0.167
Qwen3-14B (on)	balls	implied	0.75	32	1.000	0.000

Table 119: The answering position under a deceptive strategy. Deposited-item and own-answer probes (trained as in Table 113 and Table 117) read at the answering position on adv.-skip prompts, scored against the rule's oracle item and against the correct item, on problems where they differ. Accuracy at the best depth; chance is the majority-class rate of the oracle labels.

Model	Size	deposited probe		own-answer probe		chance	n
		oracle	correct	oracle	correct		
Gemma-2-9B	N=3	0.88	0.41	0.40	0.35	0.46	132
Gemma-2-9B	N=7	1.00	0.17	0.70	0.15	0.19	179
Gemma-2-9B	N=11	0.93	0.10	0.57	0.09	0.21	187
Qwen-2.5-14B	N=3	0.37	0.37	0.39	0.36	0.46	132
Qwen-2.5-14B	N=5	0.92	0.23	0.40	0.24	0.26	171
Qwen-2.5-14B	N=7	0.99	0.17	0.30	0.15	0.19	179
Qwen-2.5-14B	N=9	0.97	0.12	0.25	0.14	0.21	180
Qwen-2.5-14B	N=11	0.92	0.12	0.26	0.09	0.21	187
Qwen3-14B (off)	N=7	0.49	0.17	0.46	0.20	0.19	179
Qwen3-14B (off)	N=11	0.63	0.14	0.48	0.12	0.21	187
Qwen3-14B (on)	N=7	0.59	0.33	0.32	0.29	0.19	178
Qwen3-14B (on)	N=11	0.60	0.26	0.45	0.26	0.21	187

Table 120: Control: is the initial item readable at the answer position without the rule? The adv.-skip oracle is the queried person's initial item, which the question states. The same probes read that item on adv.-skip, correct-strategy and no-CoT prompts at the answering position ($N \geq 5$, problems where it differs from gold; each control at its own best depth).

Model	Probe	with rule	correct strategy	no CoT	chance
Gemma-2-9B	deposited	0.96	0.17	0.25	0.20
Gemma-2-9B	own answer	0.64	0.14	0.18	0.20
Qwen-2.5-14B	deposited	0.95	0.19	0.23	0.22
Qwen-2.5-14B	own answer	0.30	0.16	0.17	0.22

Table 121: Rows on which the model resisted the rule and answered correctly: what the answering position holds. Pooled over sizes, probes refitted at each cell's best depth.

Model	Probe	reads oracle	reads correct	n
Gemma-2-9B	deposited	0.00	1.00	7
Gemma-2-9B	own answer	0.14	0.43	7
Qwen-2.5-14B	deposited	0.77	0.11	349
Qwen-2.5-14B	own answer	0.34	0.17	349
Qwen3-14B (off)	deposited	0.51	0.21	79
Qwen3-14B (off)	own answer	0.34	0.24	79
Qwen3-14B (on)	deposited	0.58	0.24	335
Qwen3-14B (on)	own answer	0.36	0.26	335

Model	Position	correct item (depth)	chance	random other	shuffled p95	planted (depth)	correct, last 3 layers	n
Gemma-2-9B	answering	0.364 (0.98)	0.154	0.150	0.170	0.933 (0.88)	0.354	800
Gemma-2-9B	last prompt token	0.186 (0.52)	0.154	0.142	0.172	0.241 (0.62)	0.177	800
Llama-3.1-8B	answering	0.337 (0.97)	0.153	0.142	0.173	0.955 (0.88)	0.332	798
Llama-3.1-8B	last prompt token	0.175 (0.41)	0.154	0.142	0.169	0.206 (0.94)	0.154	798
Qwen-2.5-14B	answering	0.341 (0.96)	0.154	0.156	0.171	0.979 (0.90)	0.330	800
Qwen-2.5-14B	last prompt token	0.186 (0.42)	0.154	0.149	0.169	0.237 (0.65)	0.176	800
Qwen3-14B (off)	answering	0.220 (0.97)	0.154	0.161	0.169	0.616 (0.97)	0.212	800
Qwen3-14B (off)	last prompt token	0.284 (1.00)	0.154	0.144	0.168	0.324 (0.97)	0.251	800
Qwen3-14B (on)	answering	0.260 (1.00)	0.154	0.163	0.168	0.716 (0.75)	0.247	799
Qwen3-14B (on)	last prompt token	0.352 (1.00)	0.154	0.154	0.177	0.403 (0.97)	0.297	799

Table 122: A probe trained on corrupted prompts to read the correct item, peak accuracy over layers (depth in parentheses), against chance and two floors, next to the same probe trained to read the planted item. On the copied rows alone the correct-item peak is 0.352, 0.346, 0.343, 0.220 and 0.254 in the same order of models.

Model	depth	correct item	chance	random other	shuffled p95	gap to highest floor
Gemma-2-9B	0.50	0.159	0.169	0.116	0.155	−0.010
	0.69	0.224	0.169	0.152	0.152	+0.056
	0.86	0.238	0.169	0.153	0.157	+0.069
Llama-3.1-8B	0.50	0.176	0.158	0.120	0.153	+0.018
	0.69	0.198	0.158	0.130	0.156	+0.041
	0.84	0.317	0.158	0.134	0.156	+0.159
Qwen-2.5-14B	0.50	0.139	0.155	0.125	0.148	−0.016
	0.71	0.210	0.155	0.143	0.154	+0.055
	0.85	0.287	0.155	0.143	0.158	+0.129
Qwen3-14B (off)	0.50	0.130	0.155	0.127	0.166	−0.036
	0.70	0.160	0.155	0.160	0.155	−0.000
	0.85	0.174	0.155	0.139	0.150	+0.019
Qwen3-14B (on)	0.50	0.134	0.167	0.086	0.151	−0.033
	0.70	0.115	0.167	0.086	0.157	−0.053
	0.85	0.187	0.167	0.115	0.159	+0.019

Table 123: The correct-item probe by depth on the copied rows: at the floor at half depth, above it from about 0.7, and at its peak (Table 122) in the last two layers.

Model	Condition	rows	correct item is rank 2	median rank	median prob. (correct)	median prob. (emitted)
Gemma-2-9B	deceptive strategy	1545	0.000	20	0.0000	0.985
Gemma-2-9B	corrupted solution	1366	0.051	10	0.0000	0.992
Llama-3.1-8B	deceptive strategy	305	0.016	14	0.0001	0.981
Llama-3.1-8B	corrupted solution	948	0.020	6	0.0023	0.623
Qwen-2.5-14B	deceptive strategy	243	0.000	33	0.0000	0.970
Qwen-2.5-14B	corrupted solution	924	0.109	6	0.0000	0.991
Qwen3-14B (off)	deceptive strategy	499	0.012	26	0.0000	0.995
Qwen3-14B (off)	corrupted solution	934	0.063	10	0.0000	0.993
Qwen3-14B (on)	deceptive strategy	8	0.000	60	0.0000	0.992
Qwen3-14B (on)	corrupted solution	211	0.237	5	0.0000	1.000

Table 124: Where the correct item sits in the output distribution on the rows where the model emitted the wrong value. Rank and probability of the correct item's token, written as the model wrote the emitted one (the most probable surface variant gives the same fractions to within 0.01). Pooled over models and conditions, the correct item is rank 2 on 4.4% of copied rows and in the top five on 30%.

G.9 WHERE THE CORRECT ITEM OVERTAKES THE WRONG VALUE

Section 7 says that on the rows where the model answers correctly the correct value overtakes the planted one at about two thirds of depth. This is read with the model's own unembedding rather than with a probe, since a probe says whether an item is encoded and not which one is winning. Table 125 gives the result. On those rows the residual stream at the answering position is put through the model's final norm and unembedding at every layer (a logit lens), and L^* is the first depth after which the correct item's logit stays above the wrong value's. The lens is faithful: at the last layer its argmax is the token the model emitted on 100.0%, 99.7% and 99.8% of rows for Gemma-2, Llama-3.1 and Qwen-2.5 (90.7% and 95.9% for the two sampled Qwen3 modes, whose emitted token is a draw). Rows are kept only when the token the model writes at the anchor is the correct item and the wrong value is distinct from it (on the *swaps cancel* rule the two coincide on 437 rows, which are dropped), at $N \geq 5$. On the wider set of rows whose extracted answer is correct, the token at the anchor is the wrong value on 44.5%, the correct item on 30.7% and a letter or formatting token on 24.7%, so that set says nothing about overriding and is not used here.

Model	Condition	n	never crosses	L^* p25	median L^*	L^* p75
Gemma-2-9B	corrupted solution (item anchor)	52	0.019	0.64	0.69	0.81
Gemma-2-9B	corrupted solution	10	0.000	0.63	0.69	0.98
Llama-3.1-8B	deceptive strategy	81	0.000	0.44	0.50	0.56
Llama-3.1-8B	corrupted solution	16	0.062	0.56	0.84	0.98
Qwen-2.5-14B	deceptive strategy	8	0.000	0.68	0.71	0.72
Qwen-2.5-14B	corrupted solution	15	0.200	0.66	0.73	0.94
Qwen3-14B (off)	deceptive strategy	42	0.000	0.42	0.70	0.74
Qwen3-14B (off)	corrupted solution	73	0.014	0.55	0.71	0.75
Qwen3-14B (on)	deceptive strategy	126	0.000	0.49	0.68	0.72
Qwen3-14B (on)	corrupted solution	93	0.022	0.57	0.70	0.72
pooled		516	0.016		0.675	

Table 125: The depth at which the correct item overtakes the wrong value in the logit lens, on the rows where the model writes the correct item at the answering position. Seven of the nine cells have their median in 0.66–0.75; the hand-off band of Section 7 is 0.58–0.66. Gemma-2's corrupted-solution rows are mostly letter-first, so the item anchor (one token before the item name) is reported as well.

G.10 THINKING ON AND OFF AT THE ANSWERING POSITION

Section 8 says that with thinking on the planted item is still readable at the answering position. The Qwen3-on rows of Tables 113, 118 and 122 carry that reading (the planted item at 0.70 pooled and at 0.64 on the rows answered correctly). Table 126 adds that the representation is the same in both modes: probes trained under one Qwen3 mode and tested on rows generated under the other transfer at 0.70–0.96 for the deposited and planted items, and neither mode makes the correct item readable at the answering position beyond the faint trace of Section G.8.

Table 126: Cross-template probes at the answering position: probes trained under one Qwen3 template tested under the other ("cross") against within-template held-out accuracy ("same").

Trained on	Domain	Test	depth	acc	n
Qwen3-14B (off)	soccer	cross:corrupted/gold	0.00	0.182	132
Qwen3-14B (off)	gifts	cross:corrupted/implied	0.80	0.699	133
Qwen3-14B (off)	gifts	cross:expert_stripped/deposited	0.88	0.963	134
Qwen3-14B (off)	soccer	same:corrupted/gold	0.00	0.197	132
Qwen3-14B (off)	balls	same:corrupted/implied	1.00	0.806	134
Qwen3-14B (off)	balls	same:expert_stripped/deposited	0.85	0.799	134
Qwen3-14B (on)	gifts	cross:corrupted/gold	0.38	0.157	134
Qwen3-14B (on)	balls	cross:corrupted/implied	0.78	0.754	134
Qwen3-14B (on)	gifts	cross:expert_stripped/deposited	0.75	0.836	134
Qwen3-14B (on)	gifts	same:corrupted/gold	0.75	0.158	133
Qwen3-14B (on)	balls	same:corrupted/implied	0.78	0.724	134
Qwen3-14B (on)	balls	same:expert_stripped/deposited	0.75	1.000	134

H THINKING MODE: ADDITIONAL TABLES

This appendix holds the numbers behind the discussion on reasoner models, where the same Qwen3-14B weights are run with the thinking channel open. Section H.1 gives what the switch does to every condition, at the hardest tier and at every tier; Section H.2 gives where the work happens, in the thought or in the answer. The probe readings with thinking on, which say that the planted item is still readable at the answering position, are in Appendix G.10; the mid-derivation conditions with thinking on are in the tables of Appendix D.1.

H.1 THE SWITCH

Table 127 collects, at the hardest tier of each task, what switching Qwen3-14B into thinking mode does to correct and to wrong content: accuracy on correct content is unchanged or better, and adoption of wrong content falls from 90–100% to 14–41%. Tables 128–130 give the adoption change at every tier, which is the source of the two figures quoted in the paper: adoption with thinking on is below 5% at the easiest tiers and around 40% at the hardest ones on objects, so the resistance that thinking adds erodes as problems get harder.

	Objects, N=11		Navigate, 20 steps		Date, s_3	
What is in the prompt	off	on	off	on	off	on
Correct content — % correct						
nothing (no CoT)	93.0	99.5	94.5	100.0	73.0	93.0
the full correct trace	100.0	100.0	100.0	100.0	100.0	100.0
a correct strategy	91.0	95.5	92.5	98.0	71.0	96.0
wrong strategy + correct trace	99.5	99.0	100.0	100.0	100.0	98.5
wrong strategy alone	3.0	94.0	6.0	89.5	12.0	97.0
Wrong content — % giving the planted answer						
one wrong item in the final state	98.5	**38.5**	99.5	**32.0**	98.5	**19.5**
the same, with a wrong strategy in front	97.5	41.0	100.0	15.0	100.0	13.0
one wrong intermediate step, trace cut there	90.5	40.5	60.4	13.9	96.5	34.5
the most-adopted deceptive strategy	72.4	24.4	94.2	5.8	–	–

Table 127: **Switching the same weights into thinking mode changes what happens to wrong content and leaves correct content alone.** Qwen3-14B with thinking off and on, hardest tier of each task. *Correct content*: accuracy. *Wrong content*: how often the answer is the one the wrong content implies. The most-adopted deceptive strategy is *parity* on objects and *count turns* on navigate; date strategies are omitted because their answers are mostly not among the options. The wrong-strategy rows are the `wrong_strategy` rows of Appendix C.1.

condition	obj3	obj5	obj7	obj9	obj11
corrupted_expert	76.0 → 1.0 (-75.0)	94.5 → 7.5 (-87.0)	98.5 → 20.5 (-78.0)	99.5 → 38.5 (-61.0)	98.5 → 38.5 (-60.0)
wrong_strategy_corrupted	82.0 → 1.0 (-81.0)	96.0 → 2.5 (-93.5)	100.0 → 21.5 (-78.5)	99.0 → 32.0 (-67.0)	97.5 → 41.0 (-56.5)
mid_corrupt_stop	47.5 → 0.5 (-47.0)	82.0 → 2.0 (-80.0)	91.0 → 22.0 (-69.0)	91.5 → 32.5 (-59.0)	90.5 → 40.5 (-50.0)
adv_skip	40.2 → 0.0 (-40.2) (n=132)	54.4 → 0.0 (-54.4) (n=171)	68.7 → 0.6 (-68.2) (n=179)	64.4 → 1.7 (-62.8) (n=180)	61.5 → 2.7 (-58.8) (n=187)
adv_last_swap	17.0 → 2.0 (-15.0)	36.9 → 0.6 (-36.3) (n=168)	33.9 → 5.4 (-28.6) (n=168)	26.7 → 7.0 (-19.8) (n=172)	26.4 → 6.6 (-19.8) (n=182)
adv_parity	38.7 → 3.6 (-35.1) (n=168)	59.6 → 3.5 (-56.0) (n=141)	60.5 → 5.7 (-54.8) (n=157)	72.0 → 19.5 (-52.4) (n=164)	72.4 → 24.4 (-48.1) (n=156)

Table 128: Adoption, thinking off → on, objects.

condition	nav5	nav10	nav15	nav20
corrupted_expert	99.0 → 10.0 (-89.0)	99.5 → 17.5 (-82.0)	100.0 → 18.5 (-81.5)	99.5 → 32.0 (-67.5)
wrong_strategy_corrupted	98.5 → 7.5 (-91.0)	100.0 → 6.0 (-94.0)	100.0 → 13.5 (-86.5)	100.0 → 15.0 (-85.0)
mid_corrupt_stop	54.1 → 5.9 (-48.2) (n=85)	51.5 → 5.1 (-46.5) (n=99)	78.2 → 7.9 (-70.3) (n=101)	60.4 → 13.9 (-46.5) (n=101)
adv_skip	19.0 → 1.2 (-17.9) (n=84)	12.1 → 2.0 (-10.1) (n=99)	76.0 → 25.0 (-51.0) (n=100)	36.0 → 12.0 (-24.0) (n=100)
adv_last_step	64.1 → 12.6 (-51.5) (n=103)	100.0 → 11.3 (-88.7) (n=62)	99.1 → 21.7 (-77.4) (n=106)	100.0 → 16.7 (-83.3) (n=60)
adv_count_turns	88.2 → 22.2 (-66.0) (n=153)	100.0 → 10.4 (-89.6) (n=48)	93.5 → 28.4 (-65.2) (n=155)	94.2 → 5.8 (-88.5) (n=52)

Table 129: Adoption, thinking off → on, navigate.

condition	s1	s2	s3
corrupted_expert	98.5 → 10.0 (-88.5)	100.0 → 11.5 (-88.5)	98.5 → 19.5 (-79.0)
wrong_strategy_corrupted	99.0 → 4.5 (-94.5)	100.0 → 5.5 (-94.5)	100.0 → 13.0 (-87.0)
mid_corrupt_stop	21.0 → 1.5 (-19.5)	33.5 → 5.5 (-28.0)	96.5 → 34.5 (-62.0)

Table 130: Adoption, thinking off → on, date.

H.2 Where the work happens

Our discussion says that on a corrupted full solution the override happens in the answer and not in the thinking channel. Table 131 is the evidence: per tier, the length of the thought and of the answer with a correct solution in the prompt, the share of answers that re-derive, and the length of the whole generation when the solution is corrupted. With a solution in the prompt the thought is empty or nearly so, and when the solution is corrupted the extra work appears in the answer. Tables 132–134 give re-derivation rates with a wrong strategy in front of the correct solution, for every model, which is the comparison that shows the thinking model re-deriving where the instruct models do not.

		correct solution in the prompt			one wrong item in it
Task	tier	thought (chars)	answer (chars)	answer re-derives (%)	whole generation (chars)
Objects	N=3	0	95	3.5	1,446
	N=5	0	96	1.5	1,649
	N=7	0	90	1.0	1,890
	N=9	0	96	1.0	1,664
	N=11	0	100	0.5	2,064
Navigate	5 steps	50	558	94.0	866
	10 steps	60	900	78.0	1,117
	15 steps	50	1,137	61.0	1,338
	20 steps	60	1,392	76.5	1,444
Date	s_1	76	622	66.0	853
	s_2	75	764	99.5	1,020
	s_3	43	1,050	100.0	1,274

Table 131: **With a solution in the prompt, the thinking model does not think; whatever it does, it does in the answer.** Qwen3-14B with thinking on. Medians over 200 problems. *Thought*: characters between the `<think>` tags. With nothing in the prompt the same model thinks for 2,700–5,900 characters on objects; with the correct solution in the prompt its thought is empty on objects and under 80 characters everywhere. *Answer*: characters after `</think>`. *Answer re-derives*: share of answers that retrace at least one step of the problem rather than only stating the result. *Right column*: the same prompt with one item of the final state made wrong. On objects the generation grows 14–19×, almost all of it in the answer (980–1,680 characters), whether or not the model catches the error; the thought grows only when it does catch it (over 300 characters in 51–65% of catches, 0–1% of copies). On navigate and date, where the answer was already redoing the problem, corruption adds 100–300 characters.

model	obj3	obj5	obj7	obj9	obj11
Gemma-2-9B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Llama-3.1-8B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.5 (0.0)	0.5 (0.0)
Mistral-Small-22B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen-2.5-14B	0.0 (0.0)	1.5 (0.0)	0.5 (0.0)	0.0 (0.0)	0.5 (0.0)
Qwen3-14B OFF	2.0 (0.5)	0.5 (0.5)	0.0 (0.0)	0.0 (0.0)	1.0 (0.5)
Qwen3-14B ON	89.0 (88.5)	59.5 (59.0)	28.0 (27.5)	21.0 (20.5)	8.0 (7.0)

Table 132: Re-derivation rate with a wrong strategy in front of the correct solution, objects.

model	nav5	nav10	nav15	nav20
Gemma-2-9B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Llama-3.1-8B	0.0 (0.0) [d=194]	0.0 (0.0) [d=198]	— [d=200]	— [d=200]
Mistral-Small-22B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen-2.5-14B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen3-14B OFF	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen3-14B ON	94.0 (92.0)	84.0 (81.0)	67.0 (66.0)	81.0 (81.0)

Table 133: Re-derivation rate with a wrong strategy in front of the correct solution, navigate.

model	s1	s2	s3
Gemma-2-9B	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Llama-3.1-8B	0.0 (0.0) [d=148]	0.0 (0.0) [d=159]	0.0 (0.0) [d=138]
Mistral-Small-22B	0.0 (0.0)	0.0 (0.0)	0.6 (0.0) [d=27]
Qwen-2.5-14B	0.0 (0.0)	4.0 (3.5)	0.0 (0.0)
Qwen3-14B OFF	0.5 (0.0)	0.0 (0.0)	0.0 (0.0)
Qwen3-14B ON	64.0 (42.5)	97.5 (88.0)	100.0 (99.5)

Table 134: Re-derivation rate with a wrong strategy in front of the correct solution, date.