Model-Adaptive Tool Necessity Reveals the Knowing-Doing Gap in LLM Tool Use

Yize Cheng* **Chenrui Fan*** **Mahdi JafariRaviz*** **Keivan Rezaei** **Soheil Feizi**
University of Maryland, College Park
{yzcheng, cfan42, krezaei, mahdij, sfeizi}@umd.edu

Project: https://github.com/chengez/Tool-Cognition-Action

Abstract

Large language models (LLMs) increasingly act as autonomous agents that must decide when to answer directly vs. when to invoke external tools. Prior work studying adaptive tool use has largely treated tool necessity as a model-agnostic property, annotated by human or LLM judge, and mostly cover cases where the answer is obvious (e.g., fetching the weather vs. paraphrasing text). However, tool necessity in the wild is more nuanced due to the divergence of capability boundaries across models: a problem solvable by a strong model on its own may still require tools for a weaker one. In this work, we introduce a model-adaptive definition of tool-necessity, grounded in each model's empirical performance. Following this definition, we compare the necessity against observed tool-call behavior across four models on arithmetic and factual QA dataset, and find substantial mismatches of 26.5–54.0% and 30.8–41.8%, respectively. To diagnose the failure, we decompose tool use into two stages: an internal cognition stage that reflects whether a model believes a tool is necessary, and an execution stage that determines whether the model actually makes a tool-call action. By probing the LLM hidden states, we find that both signals are often linearly decodable, yet their probe directions become nearly orthogonal in the late-layer, last-token regime that drives the next-token action. By tracing the trajectory of samples in the two-stage process, we further discover that the majority of mismatch is concentrated in the cognition-to-action transition, not in cognition itself. These results reveal a *knowing–doing gap* in LLM tool-use: improving tool-use reliability requires not only better recognition of when tools are needed, but also better translation of that recognition into action.

1 Introduction

Large language models (LLMs) are increasingly deployed as autonomous agents that interact with external tools such as search engines, calculators, and APIs [20, 24, 26, 19]. A central challenge in building reliable autonomous LLM agents is achieving adaptive tool using: the LLM needs to determine *when* it should rely on such tools versus answering directly [8, 22, 27]. Prior work studying adaptive tool use [8, 22, 13] has largely treated tool necessity as a static, model-agnostic property, typically relying on human annotators or strong LLM judges to determine whether a query requires a tool, focusing primarily on polarized cases where the answer is obvious, such as fetching real-time weather data versus paraphrasing a static paragraph. However, tool necessity in the wild is fundamentally more nuanced due to the natural divergence of capability boundaries across different models. A problem that is easily solvable by a state-of-the-art model relying solely on its internal weights may completely exceed the capabilities of a smaller or less capable model, thereby making tool use strictly necessary for the latter but redundant for the former.

*Equal contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

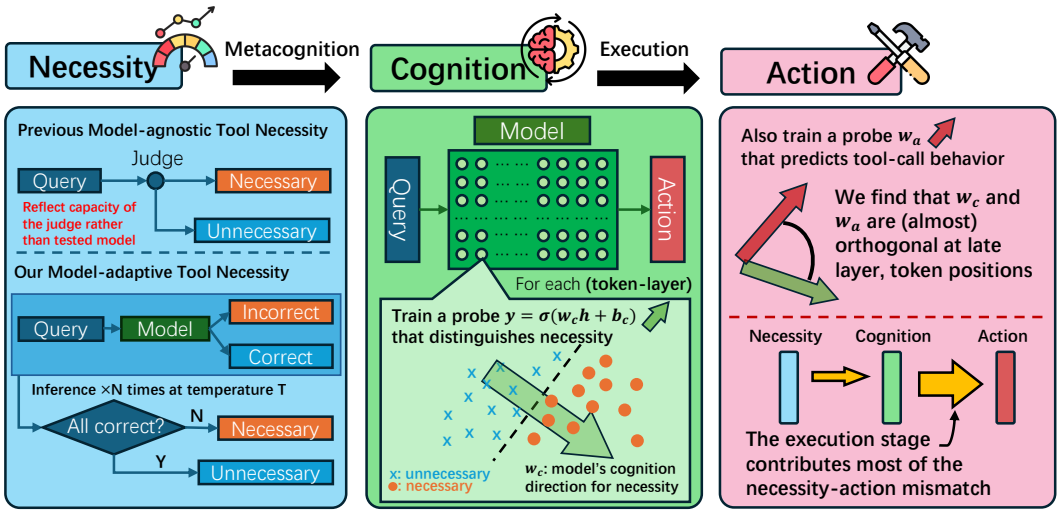


Figure 1: **Overview of the two stage cognition-execution modeling of LLM tool-use. (Left) Necessity**: We introduce a model-adaptive definition of tool necessity based on a model's empirical ability to consistently answer a query correctly on its own, contrasting with prior model-agnostic approaches. **(Middle) Cognition**: By probing the model's internal hidden states h, we identify a linear cognition direction w_c that successfully distinguishes when a tool is necessary. **(Right) Action**: We also train a probe w_a to predict the actual tool-call execution. We find that w_c and w_a become nearly orthogonal in late layers, and that the majority of the necessity-action mismatch stems from the execution stage (translating awareness into action) rather than the internal cognition stage.

In this work, we argue that tool necessity must be intrinsically tied to the specific capabilities of the model in question. We introduce a model-adaptive definition of tool necessity, grounded not in static annotations, but in each individual model's empirical performance. By evaluating necessity relative to a model's intrinsic capabilities, we establish a more accurate characterization for when a specific LLM should seek external help. Following this definition, we compare the actual necessity against the observed tool-call behavior across four distinct models on arithmetic and factual question-answering (QA) datasets. Our findings reveal substantial mismatches: models exhibit a 26.5–54.0% necessity-action mismatch in arithmetic tasks and a 30.8–41.8% necessity-action mismatch in factual QA, frequently calling tools when capable of answering directly, or attempting to answer directly when lacking the requisite internal knowledge.

To diagnose the underlying mechanisms of this failure, we propose a two-stage decomposition of the tool-use process: an internal cognition stage, which reflects whether the model's internal representations encode the belief that a tool is necessary, and an execution stage, which determines whether the model actually outputs the tool-triggering tokens. Building on prior advancements in mechanistic interpretability and representation engineering [35] and following recent literature on adaptive tool-using [13, 28], we probe the LLM hidden states and find that both the cognition of necessity and the execution intent are often linearly decodable. Yet, intriguingly, their respective probe directions become nearly orthogonal in the late-layer, last-token regime.

By tracing the trajectory of samples through this two-stage process, we uncover a *knowing-doing gap* in LLM tool use: the majority of the observed necessity-action mismatch cases originates from the transition from cognition to action, rather than in the cognition stage. Models frequently generate internal representations indicating the awareness of their own limitations, but fail to translate this into the syntactic execution of a tool call. Our main contributions can be summarized as follows:

- We introduce a model-adaptive definition of tool necessity grounded in empirical performance, challenging the traditional reliance on static, model-agnostic annotations.
- We evaluate four distinct LLMs across arithmetic and factual QA datasets, revealing substantial behavioral mismatches (up to 54.0%) between actual tool necessity and observed tool-call actions.
- By dividing tool use into an internal cognition stage and an execution stage, we use representation probing to demonstrate that while both intent and necessity are linearly decodable, their probe directions become near orthogonal in the late-layer, last-token regime.

- Through trajectory tracing, we discover that tool-use failures predominantly occur during the transition from cognition to action, highlighting a knowing-doing gap in LLM tool-use.

2 Related work

Tool calling in LLM agents. To extend LLM capabilities beyond parametric knowledge, researchers have introduced function/tool calling [20, 24, 26, 19], enabling interaction with external resources and expanding task coverage. Standardized protocols like MCP [1] and A2A [6] further streamline communication and access within tool ecosystems. In parallel, various works has examined tool-use accuracy [12, 21], hallucinated calls [33, 23], and robustness to tool descriptions [25, 5]. However, while these efforts aim at teaching and evaluating *how* to use tools, an important and often understudied challenge in building reliable LLM agents is determining *when* to use tools. Existing works that do study this challenge [8, 22, 13] treat tool necessity as a static property of the query, labeling instances as either tool-necessary or tool-unnecessary using human annotators or some proprietary LLM. This ignores the inherent difference in capability boundaries between different models. While Wang et al. [27] has also advocated for model-dependent tool necessity, to the best of our knowledge, we are the first to have a pipeline that empirically grounds tool necessity in the actual capabilities of a given model.

Meta-cognition of LLMs and the "knowing-doing gap". The ability of LLMs to accurately assess their own capability boundaries—often referred to as meta-cognition or self-assessment—has been a topic of long-standing interest [10, 30]. To measure this self-awareness, early work primarily relies on measuring explicit self-assessment by teaching models to express their knowledge boundaries [2, 31] or to directly verbalize confidence [15]. However, recent work has shown that the ability for models to verbalize its internal activations is limited [17, 9]. Moreover, the task of self-assessment and actual problem solving are fundamentally different tasks. When explicitly prompted about its capability boundary, the model would focus on self-assessment. But when faced with actual problem solving, the prompt is usually tasks-oriented, and hence the self-assessing process becomes implicit and subconscious. This akin to the distinction between system I and system II thinking [14]. Therefore, in this work, we follow some recent work that use internal state probing to measure models' cognition of tool-necessity [13, 28], and also empirically show in Appendix E how model tool-call actions change when explicitly prompted for self-assessment.

Meanwhile, papers in other domain of LLMs that leverage hidden states to study model internal cognition have found that the model's action can diverge from its internal belief. For example, Zhao et al. [34] find that LLMs may fail to refuse harmful queries despite internally recognizing their harmfulness, and Zhang et al. [32] show that models can internally recognize their inability to solve certain math problems yet still expend tokens on unproductive reasoning. In this work, we show that this "knowing-doing gap" similarly exists in tool-calling, and it can constitute even a larger proportion of end-to-end errors.

3 Defining model-adaptive tool necessity and two-stage modeling of tool-call

To study tool-use behavior in LLMs, we introduce a simple decomposition that separates recognizing the need for a tool from acting on that recognition. This distinction will serve as the foundation for the evaluation, diagnosis, and analysis throughout the rest of this paper.

Defining model-adaptive tool necessity. Existing work typically assumes a fixed notion of tool necessity, assigning each query a static label independent of the model being evaluated. However, we argue that since different models have different capability boundaries, the tool necessity label should be adaptive according to the model. To characterize a model's capability boundary, given a model f and query x, we perform N independent inference runs without access to external tools at temperature T. If the model f can consistently solve the problem x correctly across N runs, we assume that this x falls within the f's capability boundary and therefore the tool necessity, $n_f(x)$, is 0. Otherwise, the model cannot reliably solve this query, and hence $n_f(x)$ is 1. The parameters N and T control the strictness of this criterion. Specifically, larger values of N and T yield a more conservative and robust estimate of whether a query truly falls within the model's capability boundary as they demand the model to output the correct answer more consistently.

This formulation captures a key aspect of real-world deployment: reliability under uncertainty. In practical settings, a model that only occasionally produces the correct answer without tools may still benefit from external assistance to ensure consistent performance. By grounding tool necessity in empirical behavior rather than static annotation, our approach provides a more faithful characterization of when tool use is genuinely required for a given model.

The cognition-execution modeling of tool-call. We conceptualize tool use as a two-stage process:

$$x \to z_f(x) \to a_f(z_f(x)), \tag{1}$$

where $z_f(x)$ represents the model's internal cognition of whether a tool is needed, and $a_f(z_f(x))$ denotes whether the model actually invokes a tool, based on its cognition. This two-stage decomposition mirrors the cognition process of human and what we desire for the model. It distinguishes between *meta-cognition*—the model's internal belief about its capability boundary, and *execution ability*—how model acts based on its cognition.

End-to-end error diagnosis. Under our model-dependent definition of tool necessity $n_f(x)$ and the two stage modeling as in Equation 1, we can decompose the end-to-end necessity-action mismatch, $D(n_f(x), a_f(z_f(x)))$, into the mismatch between actual necessity and cognition $D(n_f(x), z_f(x))$, and the mismatch between model's cognition and actual decision $D(z_f(x), a_f(z_f(x))$, where $D(m, n)$ denotes the discrepancy between m and n.

4 Dataset curation

We cover two representative domains: math arithmetic and factual question answering, using two widely used model families: Qwen3-8B and Qwen3-4B [29], as well as Llama-3.1-8B-Instruct and Llama-3.2-3B-Instruct [7]. These domains provide natural testbeds in which some queries can be reliably solved by the model alone, while others may require external assistance (i.e. a calculator for arithmetic tasks and a search API for factual queries). For math arithmetic dataset, we mix problem types that vary in both surface form and actual difficulty. It includes simple one- and two-step addition and subtraction problems, along with harder examples involving multi-digit multiplication, modulo, parentheses, operator precedence, and longer addition/subtraction chains, resulting in a total of 4,000 instances. This gives us problems with a range of difficulty levels from very simple questions to extremely difficult ones, enabling us to measure the capability boundary of the model. More details about the curation of our arithmetic dataset can be found in Appendix A. For factual question answering, we adopt TruthfulQA [16], a widely used dataset with 817 instances designed to evaluate the factual reliability of language models.

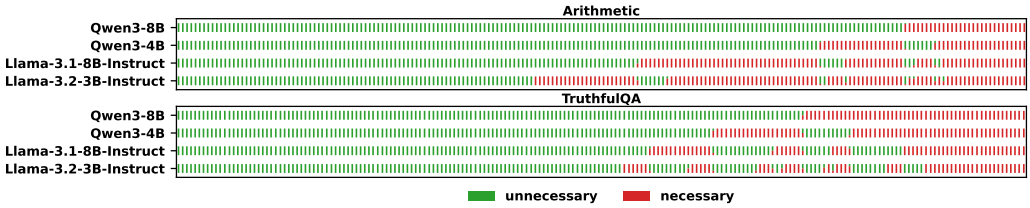


Figure 2: **Model-adaptive tool-call necessity.** Each vertical bar represents 0.5% of samples. Green indicates samples answered correctly in all $N = 10$ no-tool runs; red indicates at least one failure. Within each dataset, samples share the same order across rows, obtained by recursively sorting within each previous model's correctness partition.

4.1 Grounding tool necessity to model-specific capability boundaries

We follow our definition in Section 3 and run $N = 10$ independent inferences at temperature $T = 0.7$ without access to external tools. Sensitivity analysis under alternative N, T values are presented in Appendix B. For a specific model, we count samples where the model fails at least once as *tool-necessary*, and samples where the model consistently gives correct answers across all $N = 10$ runs as *tool-unnecessary*. Figure 2 shows that different models have substantially different

capability boundaries, which would be obscured by the model-agnostic definition of tool necessity. Specifically, the clean boundary in the first row is induced by our sorting procedure, while the red-green disagreements across rows show that the same sample groups can fall on different sides of different models' capability boundaries. This pattern appears in both arithmetic and factual question answering, suggesting that tool necessity depends not only on task type or dataset membership, but also on the particular model being deployed. This motivates using $n_f(x)$ rather than a single global necessity label when evaluating tool-use judgment and downstream call behavior.

4.2 Collecting tool-call behaviors on *tool-necessary* and *tool-unnecessary* instances

We run inference on the LLMs using both the tool-necessary and tool-unnecessary instances obtained in Section 4.1. In this setting, models are provided access to external tools: a calculator for arithmetic question answering and a search API for factual queries. To facilitate the diagnostic interpretation efforts in Section 5, greedy decoding is used when collecting tool-call actions. To better reflect real-world deployment, we follow existing practice [21, 3] and implement model-specific handlers that expose these tools in the syntax expected by each model. We then further divide *tool-necessary* and *tool-unnecessary* samples based on the model's actual tool-call behavior, and obtain 4 sets of data: *Necessary-Called* (N-C), *Necessary-NotCalled* (N-NC), *Unnecessary-Called* (UN-C), and *Unnecessary-NotCalled* (Un-NC). The first and last are aligned with the optimal behavior under our model-dependent definition of necessity, while the middle two correspond to the end-to-end necessity–action mismatch $D(n_f(x), a_f(z_f(x)))$ defined in Section 3.

Table 1: Breakdown of tool-call behavior across the four categories defined by the model-dependent necessity $n_f(x)$ and the observed action. Aligned cells (N-C: Necessary-Called, UN-NC: Unnecessary-NotCalled) are shaded green; misaligned cells (N-NC, UN-C) are shaded red and together form the end-to-end mismatch, summarized in the gray **Mis.** column.

Model	Arithmetic					TruthfulQA				
	N-C	N-NC	UN-C	UN-NC	**Mis.**	N-C	N-NC	UN-C	UN-NC	**Mis.**
Qwen3-8B	438 (11.0%)	140 (3.5%)	1526 (38.2%)	1896 (47.4%)	**41.7%**	69 (8.4%)	146 (17.9%)	108 (13.2%)	494 (60.5%)	**31.1%**
Qwen3-4B	253 (6.3%)	581 (14.5%)	481 (12.0%)	2685 (67.1%)	**26.5%**	103 (12.6%)	153 (18.7%)	189 (23.1%)	372 (45.5%)	**41.8%**
Llama-3.1-8B-Instruct	419 (10.5%)	1204 (30.1%)	335 (8.4%)	2042 (51.1%)	**38.5%**	98 (12.0%)	130 (15.9%)	122 (14.9%)	467 (57.2%)	**30.8%**
Llama-3.2-3B-Instruct	526 (13.2%)	1559 (39.0%)	600 (15.0%)	1315 (32.9%)	**54.0%**	58 (7.1%)	164 (20.1%)	104 (12.7%)	491 (60.1%)	**32.8%**

End-to-end mismatch is substantial. Table 1 reports the distribution of the four categories across four models and two domains. The aggregated mismatch rate (gray **Mis.** column) ranges from **26.5%** to **54.0%** on arithmetic and from **30.8%** to **41.8%** on TruthfulQA, indicating that with a model-specific notion of tool necessity, between roughly one quarter and one half of all queries result in a tool-use action that is inconsistent with the model's actual capability. This mismatch rate between actual tool necessity and model tool-use action further highlights the importance of determining *when* to use tools, an issue that is often overlooked in prior work that only emphasizes *how* to use them.

The dominant failure mode is highly model- and domain-dependent. Beyond the overall mismatch rates, the specific types of errors vary significantly across both models and domains. On arithmetic, Qwen3-8B suffers from tool-overuse (UN-C at **38.2%** vs. N-NC at **3.5%**). In contrast, Qwen3-4B and both Llama models exhibit clear tool underuse, with N-NC rates of **14.5%** (Qwen3-4B), **30.1%** (Llama-3.1-8B-Instruct), and **39.0%** (Llama-3.2-3B-Instruct), exceeding their respective UN-C rates. Interestingly, these tendencies are not consistent even within a single model. On TruthfulQA, Qwen3-8B reverses its trend entirely, showing tool underuse (N-NC at **17.9%** vs. UN-C at **13.2%**), while Qwen3-4B now shows tool-overuse (UN-C at **23.1%** vs. N-NC at **18.7%**). Because these models shift between being overly eager and overly conservative in tool-calling depending on the context, it is clear that no single, uniform bias can fully explain these mismatch errors. Therefore, in the next section, we leverage our two-stage modeling of LLM tool-use defined in Section 3 for more fine-grained diagnosis.

5 From meta-cognition to execution ability: What went wrong?

Having measured the model-dependent tool necessities for each model (i.e., their capability boundaries) and collected their actual tool-call behaviors, we now examine where the breakdown between

actual necessity and final action occurs, following the two-stage decomposition in Section 3. We first show that each stage—the internal cognition of necessity, and the executed action—is individually linearly separable from the model's hidden states (Section 5.1 and Section 5.2), and then characterize the geometric relationship between the two (Section 5.3). Finally, we find that the majority of the error originates in the execution stage through per sample tracing (Section 5.4).

5.1 Probing for model's cognition

Linear probing is a standard method for studying how concepts are represented in a model's hidden-state space. Recent works [13, 28] have used it as a proxy for models' internal belief of tool necessity and reported that, despite substantial end-to-end mismatch, the hidden states of *tool-necessary* and *tool-unnecessary* samples are *almost linearly separable*. Because that conclusion was drawn under a static, query-only definition of tool necessity, it is unclear whether it survives the model-dependent definition introduced in Section 3, where the necessity label $n_f(x)$ varies across models with different capability boundaries.

Concretely, we train a linear classifier with weight $\mathbf{w}_c$ and bias b_c on the model's hidden states, using a learning rate of 0.01 with the Adam [11] optimizer, minimizing the following objective:

$$\mathcal{L} = -\frac{1}{K}\sum_{k=1}^{K}\left[n_f(x_k)\log\sigma(\mathbf{w}_c^\top h_t^{(l)}(x_k)+b_c) + (1-n_f(x_k))\log(1-\sigma(\mathbf{w}_c^\top h_t^{(l)}(x_k)+b_c))\right], \tag{2}$$

where x_k is a sample in the dataset and $h_t^{(l)}(x_k)$ is the hidden state at token position t and layer l. $\mathbf{w}_c$ also serves as the normal vector of the separating hyperplane, indicating the direction from "unnecessary" to "necessary" in the model's representation space. We sweep (t, l) over all layers and over the last 20 query tokens; negative indices denote token positions relative to the start of generation, e.g., $t = -1$ is the final query token. As the class distribution is imbalanced (Table 1), we report the probe performance using the Matthews Correlation Coefficient (MCC) [18] on the held-out test set (30% of data), which is a more robust metric than accuracy or F1 under skewed labels:

$$\text{MCC} = \frac{TP\cdot TN - FP\cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{3}$$

Typically, an MCC value between 0.3-0.5 is considered moderate to good performance, and an MCC of 0.5 or more is considered good to strong performance. Figure 3 shows the MCC of probes trained at each (t, l) position for all four models on Arithmetic and TruthfulQA.

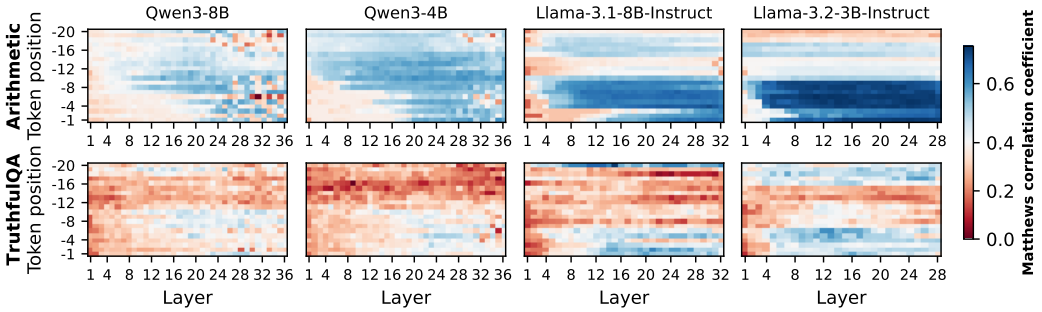


Figure 3: **Necessity probe performance across token-layer positions.** Each cell reports the held-out MCC of a linear probe trained to predict the model-adaptive necessity from the hidden state at a given layer and token position. Darker blue indicates stronger linear separability. The linear separability is strongly task-dependent, and the heatmap structure appears similar within model families.

Linear separability of necessity is strongly task-dependent. Under our model-adaptive definition, the prior "almost linearly separable" picture partially holds. On Arithmetic, necessity is linearly separable for most models, with broad regions of mid-to-late layers crossing $\text{MCC} = 0.4$. This aligns with the finding in prior works [13, 28]. On TruthfulQA, however, the regions where MCC exceeds 0.4 is noticeably smaller, with only near-last tokens in mid-late layers of Llama models still

display decent separability. This contrast suggests the challenge in distinguishing model-adaptive *tool-necessary* and *tool-unnecessary* samples, which is more nuanced than the obvious cases prior work focus on [8, 13, 28]. It also suggests that tool-necessity signals are easier to surface in tasks where problem difficulty is reflected in the input's surface structure, such as arithmetic, where complexity grows with the expression itself. In open-domain factual QA, however, surface form provides little cue about underlying difficulty, making tool necessity or epistemic uncertainty harder to linearly separate. The heatmap structure also appears similar within model families, with two Qwen and two Llama models sharing similar patterns respsectively.

Decent internal signal coexists with large end-to-end mismatch. The probe reaches decent MCC at many (t, l) positions, indicating that information about the model's capability boundary is in fact present in the residual stream. Yet the same models still exhibit substantial end-to-end necessity–action mismatch (Table 1), meaning this internal signal is not effectively converted into the right tool-call decision at generation time. This mismatch between "what the hidden states know" and "what the model does" is a first hint of a knowing–doing gap, and motivates the next two questions: does the model encode its action in a similarly separable way (Section 5.2), and how does the action representation relate to the cognition representation (Section 5.3)?

5.2 Probing for action

Having characterized how necessity is represented internally, we now ask the parallel question for the model's actual decision: how linearly separable is the executed action—whether the model invokes a tool or not—from the same hidden states? Concretely, we train a linear classifier $(\mathbf{w}_a, b_a)$ with the same objective as in Equation 2 while just changing $(\mathbf{w}_c, b_c)$ to $(\mathbf{w}_a, b_a)$. The Probe performance on the held-out test set (30% of data) in terms of MCC is shown in Figure 4.

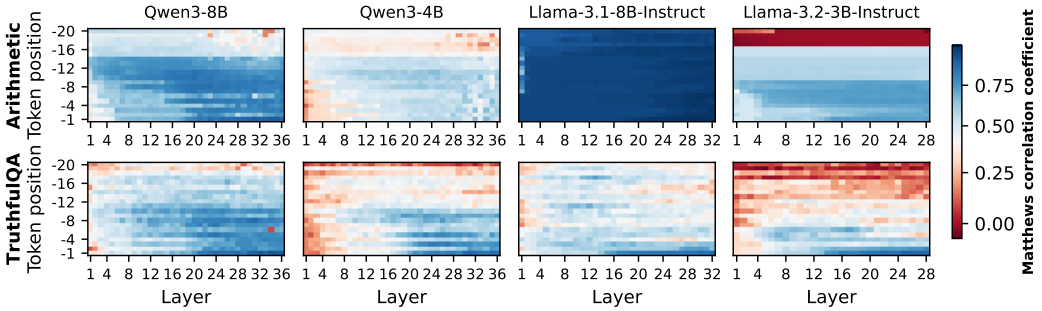


Figure 4: **The action Probe performance on different position in Matthews correlation coefficient.** Each cell reports the held-out MCC of a linear probe trained to predict the tool-call action from the hidden state at a given layer and token position. Darker blue indicates stronger linear separability. The action signal appears highly linearly separable in the hidden states, particularly in near-end tokens and late layers.

The action is highly separable from hidden states. Figure 4 shows that, on both Arithmetic and TruthfulQA dataset, the action probe attains $\mathrm{MCC} \geq 0.4$ over broad regions of nearly every model. The signal spans most layers and token positions rather than being confined to a narrow band, indicating that whether the model is about to call a tool is a strongly decodable feature from its residual stream, aligning with recent finding [4]. In Appendix C, we further show that one can causally flip the action via activation steering.

5.3 The gap between cognition and execution

The two probes give us, at every (t, l), a pair of direction vectors: $\mathbf{w}_c$ pointing from "unnecessary" to "necessary" in the model's representation space, and $\mathbf{w}_a$ pointing from "no-call" to "call." If tool-use behavior were a direct readout of the model's internal necessity assessment, the two directions should align—at least in the layers where both probes succeed. We test this by computing the cosine similarity $\mathrm{CosSim}(\mathbf{w}_c, \mathbf{w}_a)$ between $\mathbf{w}_c$ and $\mathbf{w}_a$ at each position: a value near ± 1 means necessity

and action are encoded along (anti-)parallel directions, while a value near 0 means the two are represented in geometrically independent subspaces.

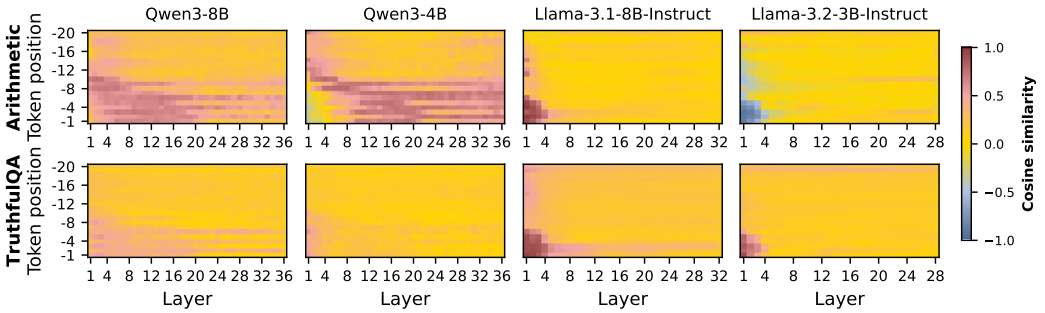


Figure 5: **The cosine similarity score between $\mathbf{w_c}$ and $\mathbf{w_a}$ on different positions.** The similarity scores between two probe direction are small across the majority of the area. Although for some models there are moderate similarity scores in late token and middle layer position, two directions fall back to near orthogonal relationship in the late layer of the last token (bottom right corner).

Partial alignment between $\mathbf{w}_c$ and $\mathbf{w}_a$ exists in intermediate token-layer positions. Figure 5 shows that the cosine similarity between $\mathbf{w}_c$ and $\mathbf{w}_a$ is fairly high in notable regions of several heatmaps, particularly covering a sizable area for the two Qwen models on Arithmetic. So necessity and action are not encoded in entirely disjoint subspaces: in some intermediate token-layer positions, the directions share meaningful alignment.

Alignment collapses at the position that drives generation. The picture changes sharply at the position that actually determines the next token: the late layers of the final query token ($t = -1$, large l). For the two models with the strongest mid-stream alignment, Qwen3-8B and Qwen3-4B, the cosine similarity falls back to small values exactly in the bottom right corner of the heatmap, so $\mathbf{w}_c$ and $\mathbf{w}_a$ become close to orthogonal precisely where they would need to interact to translate "I should call a tool" into the actual call token. The same trend toward low cosine at late layers / last token holds, more uniformly, for the other models and for TruthfulQA. Whatever partial coupling exists in earlier layers therefore does not survive to the readout. In Appendix D, we further show that this decoupling effect also exists in models with much larger scale.

The previous two subsections established that the model's hidden states often contain a usable necessity signal yet still produce mismatched tool-call actions. Figure 5 explains *why*: even when necessity and action share some structure in intermediate representations, the two directions become nearly orthogonal in the late-layer / last-token regime that ultimately drives the next-token decision.

5.4 Two stage error diagnosis and attribution

So far we have established two facts: end-to-end necessity–action mismatch is substantial (Table 1), and the cognition and action directions are nearly orthogonal at the readout (Section 5.3). However, the results in Section 5.3 tells us only that the two stages are *decoupled*, not which of them is responsible for the mismatch we observe. To attribute the error, we trace each sample along the *Factual* $\rightarrow$ *Cognition* $\rightarrow$ *Action* modeling, taking Cognition to be the necessity probe $(\mathbf{w}_c, b_c)$ read out at the last query token and last layer—the same position that drives the next-token decision. Each sample then falls into one of four categories: correct in both stages (green), stage-one-only error (red), stage-two-only error (orange, the knowing–doing gap), or compensating errors that cancel at the action (purple). We show the full Sankey flow diagram in Figure 6.

Stage two carries the majority of error. In all four panels of Figure 6, the orange flow (samples with stage two error only) is by far the largest error category, while red (samples with stage one error only) is rather thin. Given that cognition and action are decoupled, the asymmetric orange $\gg$ red localizes the failure: the end-to-end mismatch is overwhelmingly produced in the cognition $\rightarrow$ action stage rather than in forming cognition itself. The bottleneck is therefore not knowing whether a tool is needed, but converting that knowledge into the call/no-call action. This shows that making correct *when* to use tools decisions is not just about having the correct tool-necessity cognition, but more importantly also about translating that cognition to actual matching action.

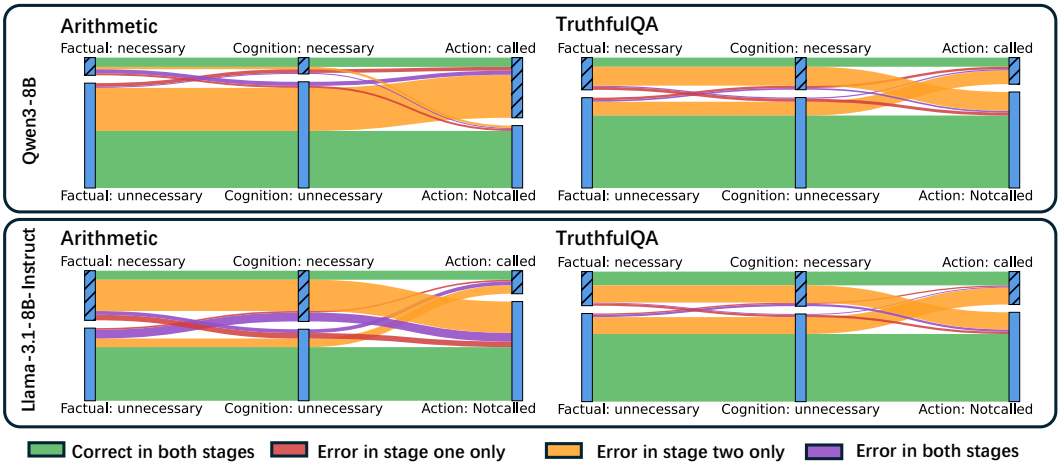


Figure 6: **Per-sample two-stage decomposition of tool-call behavior on Arithmetic and TruthfulQA, for Qwen3-8B (top) and Llama-3.1-8B-Instruct (bottom).** Each flow tracks a sample through three nodes: ground-truth necessity (*Factual*), the model's internal cognition of necessity (*Cognition*), and the executed action (*Action*). The end-to-end error is dominated by **orange** flow, where cognition is correct but action flips away from it—the knowing–doing gap.

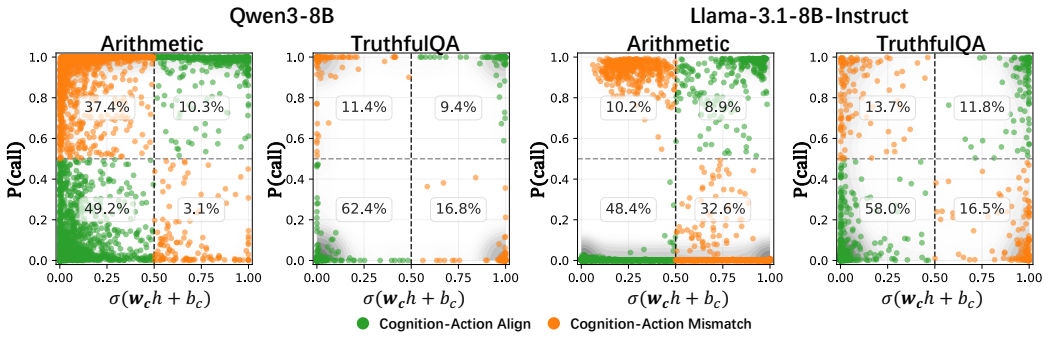


Figure 7: **The confidence of cognition versus tool calling behavior.** The x axis denotes the probe output after sigmoid function, and the y axis represents the tool call probability quantified by Equation 4. The mismatch can persist even when the internal representation strongly indicates that a sample is either *tool-necessary* or *tool-unnecessary*.

The cognition–execution mismatch is not associated with cognition confidence. Given the substantial gap between a model's internal cognition and its final tool-call behavior, illustrated by the large orange band in Figure 6, a natural question is whether this mismatch is caused by uncertainty in the meta-cognitive belief itself. In other words, does the mismatch primarily occur on samples where the model is uncertain about whether a tool is necessary? To investigate this question, we quantify the confidence of meta-cognition using the post-sigmoid output of the cognition probe, $\sigma(\mathbf{w}_c h + b_c)$. We then plot, for all samples, the relationship between the "confidence of tool necessity" and the "probability of making a tool call" in Figure 7. The probability of making a tool call is defined as

$$\mathrm{P}(\text{call}) = \frac{\mathrm{p}(\langle\text{tool-token}\rangle)}{\mathrm{p}(\langle\text{tool-token}\rangle) + \mathrm{p}(\text{best non-tool token})}, \tag{4}$$

where $p(\cdot)$ denotes the softmax probability assigned by the language model to a candidate next token, $\langle$tool-token$\rangle$ denotes the model-specific token that initiates a tool call (e.g., `<tool_call>` in Qwen models), and the "best non-tool token" refers to the highest-logit token among all tokens that are not tool-call tokens. This formulation nicely normalizes the probability to the range $[0, 1]$. Since greedy decoding is used when collecting tool-call behaviors (Section 4.2), a value of $P(\text{call}) > 0.5$ corresponds to an actual tool call being generated. As shown in Figure 7, the cognition–execution

mismatch does not primarily occur near the uncertain region where $\sigma(\mathbf{w}_c h + b_c) \approx 0.5$. Instead, many orange points occur in regions where $\sigma(\mathbf{w}_c h + b_c)$ is close to 0 or 1. This observation suggests that the cognition–execution mismatch is not driven by low confidence in the model's internal cognition. Rather, the mismatch can persist even when the internal representation strongly indicates that a sample is either *tool-necessary* or *tool-unnecessary*.

What can we do about it? A natural question arising from our diagnosis is how to address the identified knowing-doing gap. The most straightforward mitigation solution that naturally extends from our results is to use the trained necessity probe as a latent tool-necessity estimator during inference and adjust the model's tool-calling behavior on the fly, either through activation steering or guided decoding. This directly targets both types of misaligned tool-use cases: *Necessary-NotCalled* and *Unnecessary-Called*.

Since the probe is trained on the training split, evaluating the mitigation on that same split would not be meaningful. We therefore conduct additional mitigation experiments on Qwen3-8B and Llama-3.1-8B-Instruct using held-out test sets. When the probe predicts "necessary," we apply guided decoding to force the tool-call token and trigger a tool call; when the probe predicts "unnecessary," we instead mask the tool-call token to force a direct answer. The results are shown in Table 2.

Table 2: Mitigation of misaligned tool use through probe-guided decoding on held-out test sets. Misaligned rates are reported before and after mitigation, along with relative reductions.

Model	Dataset	Misaligned Rate Before	Misaligned Rate After	Relative Reduction
Qwen3-8B	Arithmetic	41.58%	9.42%	77.35%
	TruthfulQA	30.49%	19.92%	34.67%
Llama-3.1-8B-Instruct	Arithmetic	37.83%	14.75%	61.01%
	TruthfulQA	28.66%	15.85%	44.68%

Across both models and datasets, leveraging the necessity probe as an inference-time control signal consistently reduces misaligned tool use, with relative reductions ranging from 34.67% to 77.35%. These results provide a simple and concrete mitigation strategy that naturally follows from our analysis, demonstrating that the latent necessity signal identified in our work can be operationalized to improve tool-calling behavior on unseen examples.

6 Conclusion

In this work, we introduced a model-adaptive definition of tool necessity that grounds evaluation in empirical capabilities, and revealed a substantial mismatch between when models actually need tools and when they invoke them. By decomposing the tool-use process into internal cognition and execution stages, and analyzing hidden state representations, we identified a fundamental "knowing-doing gap" in LLMs. While models sometimes internally recognize the tool necessity, these cognitive representations become orthogonally misaligned with execution intent in later layers, leading to failures in taking the appropriate action. Our findings demonstrate that improving autonomous agents requires not just better internal meta-cognition, but bridging the knowing-doing gap to ensure self-awareness translates into reliable execution.

Acknowledgments

This work was supported in part by NSF CAREER Award 1942230, the ONR PECASE Award N00014-25-1-2378, ARO Early Career Program Award 310902-00001, Army Grant W911NF-21-2-0076, NSF Award CCF-2212458, NSF Award 2229885 (NSF Institute for Trustworthy AI in Law and Society, TRAILS), MURI Grant 14262683, DARPA AIQ Grant HR00112590066, and a Meta Research Award 314593-00001.

References

[1] Anthropic. Introducing the model context protocol, November 2024. URL https://www.anthropic.com/news/model-context-protocol.

[2] Lida Chen, Zujie Liang, Xintao Wang, Jiaqing Liang, Yanghua Xiao, Feng Wei, Jinglei Chen, Zhenghong Hao, Bing Han, and Wei Wang. Teaching large language models to express knowledge boundary from their own signals, 2024. URL https://arxiv.org/abs/2406.10881.

[3] Yize Cheng, Arshia Soltani Moakhar, Chenrui Fan, Parsa Hosseini, Kazem Faghih, Zahra Sodagar, Wenxiao Wang, and Soheil Feizi. Your LLM agents are temporally blind: The misalignment between tool use decisions and human time perception. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Findings of the Association for Computational Linguistics: ACL 2026*, pages 37082–37104, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1848. URL https://aclanthology.org/2026.findings-acl.1848/.

[4] Esakkivel Esakkiraja, Sai Rajeswar, Denis Akhiyarov, and Rajagopal Venkatesaramani. Therefore i am. i think, 2026. URL https://arxiv.org/abs/2604.01202.

[5] Kazem Faghih, Wenxiao Wang, Yize Cheng, Siddhant Bharti, Gaurang Sriramanan, Sriram Balasubramanian, Parsa Hosseini, and Soheil Feizi. Tool preferences in agentic LLMs are unreliable. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20954–20969, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1060. URL https://aclanthology.org/2025.emnlp-main.1060/.

[6] Google. Agent2agent (a2a) protocol. https://google.github.io/A2A/, 2025.

[7] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[8] Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. MetaTool benchmark for large language models: Deciding whether to use tools and which to use, 2024. URL https://arxiv.org/abs/2310.03128.

[9] Li Ji-An, Hua-Dong Xiong, Robert C. Wilson, Marcelo G. Mattar, and Marcus K. Benna. Language models are capable of metacognitive monitoring and control of their internal activations, 2025. URL https://arxiv.org/abs/2505.13763.

[10] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[12] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023.

[13] Wenjun Li, Dexun Li, Kuicai Dong, Cong Zhang, Hao Zhang, Weiwen Liu, Yasheng Wang, Ruiming Tang, and Yong Liu. Adaptive tool use in large language models with meta-cognition trigger. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13346–13370, 2025.

[14] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Xiao Liang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models, 2025. URL https://arxiv.org/abs/2502.17419.

[15] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words, 2022. URL https://arxiv.org/abs/2205.14334.

[16] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. URL https://arxiv.org/abs/2109.07958.

[17] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

[18] B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451, 1975. ISSN 0005-2795. doi: https://doi.org/10.1016/0005-2795(75)90109-9. URL https://www.sciencedirect.com/science/article/pii/0005279575901099.

[19] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

[20] Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.

[21] Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[22] Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. SMART: Self-aware agent for tool overuse mitigation, 2025. URL https://arxiv.org/abs/2502.11435.

[23] Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. When2call: When (not) to call tools. *arXiv preprint arXiv:2504.18851*, 2025.

[24] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36: 68539–68551, 2023.

[25] Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Prompt injection attack to tool selection in llm agents. *arXiv preprint arXiv:2504.19793*, 2025.

[26] Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, et al. Restgpt: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*, 2023.

[27] Hongru Wang, Cheng Qian, Manling Li, Jiahao Qiu, Boyang Xue, Mengdi Wang, Heng Ji, Amos Storkey, and Kam-Fai Wong. Position: Agent should invoke external tools only when epistemically necessary, 2026. URL https://arxiv.org/abs/2506.00886.

[28] Youjin Wang, Run Zhou, Rong Fu, Shuaishuai Cao, Hongwei Zeng, Jiaxuan Lu, Sicheng Fan, Jiaqiao Zhao, and Liangming Pan. ASA: Training-free representation engineering for tool-calling agents, 2026. URL https://arxiv.org/abs/2602.04935.

[29] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[30] Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. Do large language models know what they don't know? In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.551. URL https://aclanthology.org/2023.findings-acl.551/.

[31] Hanning Zhang, Shizhe Diao, Yong Lin, Yi R. Fung, Qing Lian, Xingyao Wang, Yangyi Chen, Heng Ji, and Tong Zhang. R-tuning: Instructing large language models to say 'i don't know', 2024. URL https://arxiv.org/abs/2311.09677.

[32] Qingjie Zhang, Yujia Fu, Yang Wang, Liu Yan, Tao Wei, Ke Xu, Minlie Huang, and Han Qiu. Stop before you fail: Operational capability boundaries for mitigating unproductive reasoning in large reasoning models, 2026. URL https://arxiv.org/abs/2509.24711.

[33] Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, et al. Toolbehonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. *arXiv preprint arXiv:2406.20015*, 2024.

[34] Jiachen Zhao, Jing Huang, Zhengxuan Wu, David Bau, and Weiyan Shi. LLMs encode harmfulness and refusal separately, 2025. URL https://arxiv.org/abs/2507.11878.

[35] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2025. URL https://arxiv.org/abs/2310.01405.

A More details on arithmetic dataset curation

We generate math arithmetic problems grouped into three types. The easy group contains one-step addition/subtraction, short two-step chains, and small modulo problems. These give cases where a calculator is usually not needed. The larger short group keeps the expressions simple, but uses larger operands, including multi-digit subtraction, four-digit addition/subtraction, and two- or three-digit multiplication. These problems are still short enough to invite a direct answer, but they are more likely to cause digit errors. The multi-step group contains precedence chains, parenthesized expressions, multiplication chains, and long addition/subtraction chains. These examples test whether the model can track intermediate values and apply the order of operations, especially in cases that look simple but are easy to miscompute. We sample the dataset with a fixed random seed. During generation, we skip repeated expressions and resample until each family reaches its assigned sampling share. Table 3 gives the problem families and their sampling shares.

Table 3: Breakdown of the math arithmetic dataset. Shares show the fixed sampling share for each problem family. We use 4000 samples overall.

Group	Problem Family	Share	Detail	Example
Easy	Single-step arithmetic	8%	Two operands from 1–99 with an addition or subtraction operator.	`21 + 59`
	Two-step arithmetic	5%	Three operands from 1–99 with addition or subtraction operators.	`70 - 47 + 68`
	Small modulo	5%	Three-digit dividend with divisor from 3–19.	`562 % 8`
Larger short	Negative subtraction	7%	Three- or four-digit operand minus a larger operand.	`390 - 554`
	Four-digit addition/subtraction	6%	Two four-digit operands.	`4921 - 9108`
	Two-digit multiplication	9%	Two two-digit operands.	`34 * 75`
	Three-by-two multiplication	9%	One three-digit factor and one two-digit.	`504 * 61`
	Three-by-three multiplication	7%	Two three-digit factors.	`867 * 671`
Multi-step	Precedence chain	12%	Five two- or three-digit operands with addition, subtraction, or multiplication operators.	`84 * 82 - 755 - 805 - 29`
	One-digit addition/subtraction chain	11%	16–39 one-digit terms with addition or subtraction operators.	`3 + 4 + 1 + 3 - 8 - ...`
	Small addition/subtraction chain	10%	21–27 terms from 1–30 with addition or subtraction operators.	`9 + 5 - 1 - 3 + 23 + ...`
	Parenthesized expression	6%	Four two-digit operands in $(a+b)\times(c-d)$ form.	`(67 + 68) * (52 - 88)`
	Multiplication chain	5%	Five two-digit operands in $a+b\times c-d\times e$ form.	`94 + 40 * 50 - 24 * 87`

Algorithms 1–13 specify the exact procedures used for generating the data samples in each family. In these algorithms, $\mathcal{U}\{m,\dots,n\}$ denotes the discrete uniform distribution over integers from m to n, and $\mathcal{U}(0,1)$ denotes the continuous uniform distribution on the unit interval.

Algorithm 1 SINGLESTEPARITHMETIC

$a, b \sim \mathcal{U}\{1,\dots,99\}$
$op \sim \{+,-\}$
return "a op b"

Algorithm 2 TWOSTEPARITHMETIC

$a, b, c \sim \mathcal{U}\{1,\dots,99\}$
$u \sim \mathcal{U}(0,1)$
if $u < 0.5$ **then**
    return "$a + b - c$"
else
    return "$a - b + c$"
end if

Algorithm 3 SMALLMODULO

$a \sim \mathcal{U}\{100, \ldots, 999\}$
$b \sim \mathcal{U}\{3, \ldots, 19\}$
return "$a \ \% \ b$"

Algorithm 4 NEGATIVESUBTRACTION

$u \sim \mathcal{U}(0, 1)$
if $u < 0.55$ **then**
    $a \sim \mathcal{U}\{100, \ldots, 500\}$
    $b \sim \mathcal{U}\{a + 10, \ldots, a + 250\}$
else
    $a \sim \mathcal{U}\{1000, \ldots, 5000\}$
    $b \sim \mathcal{U}\{a + 100, \ldots, a + 3000\}$
end if
return "$a - b$"

Algorithm 5 FOURDIGITADDITIONSUBTRACTION

$a, b \sim \mathcal{U}\{1000, \ldots, 9999\}$
$u \sim \mathcal{U}(0, 1)$
if $u < 0.6$ **then**
    return "$a + b$"
else
    return "$a - b$"
end if

Algorithm 6 TWODIGITMULTIPLICATION

$u \sim \mathcal{U}(0, 1)$
if $u < 0.45$ **then**
    $a, b \sim \mathcal{U}\{15, \ldots, 50\}$
else
    $a, b \sim \mathcal{U}\{30, \ldots, 99\}$
end if
return "$a \times b$"

Algorithm 7 THREEBYTWOMULTIPLICATION

$a \sim \mathcal{U}\{100, \ldots, 999\}$
$b \sim \mathcal{U}\{10, \ldots, 99\}$
return "$a \times b$"

Algorithm 8 THREEBYTHREEMULTIPLICATION

$a, b \sim \mathcal{U}\{100, \ldots, 999\}$
return "$a \times b$"

Algorithm 9 PRECEDENCECHAIN

$a_1, \ldots, a_5 \sim \mathcal{U}\{10, \ldots, 999\}$
$op_1, \ldots, op_4 \sim \{+, -, \times\}$
return "$a_1 \ op_1 \ a_2 \ op_2 \ a_3 \ op_3 \ a_4 \ op_4 \ a_5$"

Algorithm 10 ONEDIGITADDITIONSUBTRACTIONCHAIN

$u \sim \mathcal{U}(0, 1)$
if $u < 0.4$ **then**
    $n \sim \mathcal{U}\{16, \ldots, 22\}$
else
    $n \sim \mathcal{U}\{29, \ldots, 39\}$
end if
$a_1, \ldots, a_n \sim \mathcal{U}\{1, \ldots, 9\}$
For each i, sample op_i with $\Pr(op_i = +) = 0.53$ and $\Pr(op_i = -) = 0.47$
return "$a_1\ op_1\ a_2\ op_2\ \cdots\ op_{n-1}\ a_n$"

Algorithm 11 SMALLADDITIONSUBTRACTIONCHAIN

$n \sim \mathcal{U}\{21, \ldots, 27\}$
$a_1, \ldots, a_n \sim \mathcal{U}\{1, \ldots, 30\}$
$op_1, \ldots, op_{n-1} \sim \{+, -\}$
return "$a_1\ op_1\ a_2\ op_2\ \cdots\ op_{n-1}\ a_n$"

Algorithm 12 PARENTHESIZEDEXPRESSION

$a, b, c, d \sim \mathcal{U}\{10, \ldots, 99\}$
return "$(a + b) \times (c - d)$"

Algorithm 13 MULTIPLICATIONCHAIN

$a, b, c, d, e \sim \mathcal{U}\{10, \ldots, 99\}$
return "$a + b \times c - d \times e$"

B Sensitivity analysis under alternative N, T values

We conducted sensitivity analysis using $N \in \{5, 10, 15\}$ at $T = 0.7$ and $T \in \{0.5, 0.7, 0.9\}$ at $N = 10$, across both datasets and all four models. As shown in Table 4, increasing N consistently increased the proportion of tool-necessary samples in every dataset-model combination: averaged across models, the proportion increased from 22.5% to 28.2% to 31.4% on TruthfulQA, and from 26.2% to 32.0% to 34.7% on arithmetic. Increasing T produced the same overall tendency, although the effect was smaller: the average tool-necessary proportion changed from 28.0% to 28.2% to 28.4% on TruthfulQA and from 29.9% to 32.0% to 33.3% on arithmetic. These results confirm the intended behavior: stricter settings generally classify more queries as tool-necessary, while the paper's qualitative conclusions are not tied to selecting specific N/T configurations.

C Causally flipping actions via activation steering on the action probe

To investigate causality, we conduct activation steering to examine whether intervening on the model's hidden activations can flip its tool-use actions. Specifically, we intervene on the model's hidden activations at inference time and test whether moving the representation along the action probe direction changes the resulting tool-use action.

We evaluate Llama-3.1-8B-Instruct and Qwen3-8B on both Arithmetic and TruthfulQA, applying steering to the originally misaligned cases: positive steering is applied to Necessary-NotCalled examples to induce a tool call, while negative steering is applied to Unnecessary-Called examples to suppress an unnecessary call. Thus, a successful flip corresponds to correcting the model's original behavior.

After sweeping over a few possible steering strengths, we obtain the corrective flip rates shown in Table 5. The results demonstrate that steering along the action probe direction can effectively alter

Table 4: Sensitivity analysis under alternative N and T values across TruthfulQA and arithmetic. Entries report the percentage of prompts labeled tool-necessary. The Average column reports the mean across the four models.

(a) Fixed $T = 0.7$: varying N						
Dataset	N	**Average**	**Llama-3.1-8B**	**Llama-3.2-3B**	**Qwen3-4B**	**Qwen3-8B**
TruthfulQA	5	**22.5%**	20.9%	20.0%	26.6%	22.6%
	10	**28.2%**	27.9%	27.2%	31.3%	26.3%
	15	**31.4%**	32.6%	33.3%	32.4%	27.1%
Arithmetic	5	**26.2%**	33.9%	46.7%	13.9%	10.2%
	10	**32.0%**	40.6%	52.1%	20.9%	14.4%
	15	**34.7%**	43.8%	53.9%	24.6%	16.3%
(b) Fixed $N = 10$: varying T						
Dataset	T	**Average**	**Llama-3.1-8B**	**Llama-3.2-3B**	**Qwen3-4B**	**Qwen3-8B**
TruthfulQA	0.5	**28.0%**	27.4%	28.5%	29.6%	26.4%
	0.7	**28.2%**	27.9%	27.2%	31.3%	26.3%
	0.9	**28.4%**	27.4%	27.9%	31.3%	27.1%
Arithmetic	0.5	**29.9%**	37.4%	49.6%	19.6%	13.1%
	0.7	**32.0%**	40.6%	52.1%	20.9%	14.4%
	0.9	**33.3%**	43.5%	53.7%	21.8%	14.2%

the model's tool-use decisions, providing causal evidence that the probed action representation is involved in the model's tool-use behavior. Such steering would not be meaningful on the cognition probe as only the action is directly observable from the model output. There exists no observable model output that can faithfully represent a model's internal cognition, hence the probing result on the cognition probe remains correlational.

Table 5: Corrective action flip rates (%) under activation steering along the action probe direction. Negative steering is applied to Unnecessary-Called examples to suppress unnecessary tool calls, while positive steering is applied to Necessary-NotCalled examples to induce necessary tool calls.

Dataset	**Model**	**Suppress unnecessary calls (negative direction)**	**Induce necessary calls (positive direction)**
Arithmetic	Llama-3.1-8B-Instruct	100.0%	100.0%
	Qwen3-8B	94.5%	100.0%
TruthfulQA	Llama-3.1-8B-Instruct	99.1%	98.4%
	Qwen3-8B	97.0%	66.2%

D Generalization to larger models

To test whether the knowing–doing gap persists at a substantially larger model scale, we repeat our probing analysis on Llama-3.1-70B-Instruct [7]. We use the same definition of model-adaptive necessity labels as described in Section 4.1 and linear-probe protocol as in Sections 5.1 and 5.2. For each dataset, we train a cognition probe to predict whether tool use is necessary for the model and an action probe to predict whether the model calls the tool. We report held-out Matthews correlation coefficient (MCC) and compute the cosine similarity between the two probes' weight vectors at each token–layer position. The 70B activations cover all 80 transformer layers and the final five query tokens.

Necessity remains decodable at 70B, with the same task dependence. Figure 8 shows a broad and stable necessity signal on Arithmetic: 86.25% of the evaluated token–layer positions attain cognition-probe MCC ≥ 0.4. The signal is weaker and more localized on TruthfulQA, where 39.50% of positions cross the same threshold. Nevertheless, the meta-cognitive tool-necessity signal remains

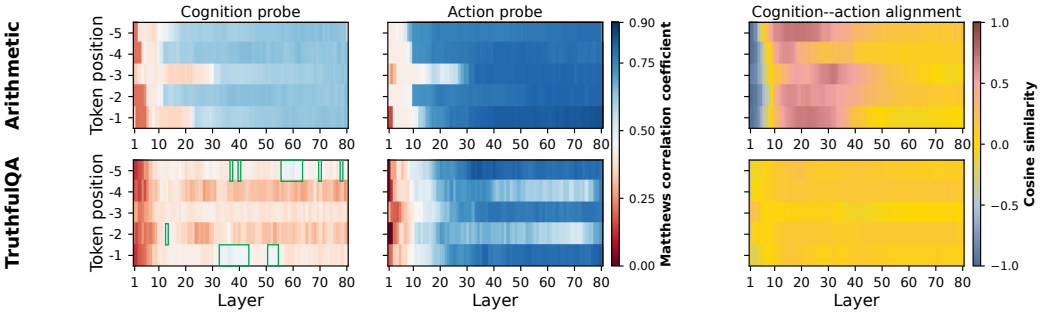


Figure 8: **Cognition, action, and cognition–action alignment for Llama-3.1-70B-Instruct.** Rows show Arithmetic (top) and TruthfulQA (bottom); columns show the held-out MCC of the cognition (necessity) probe, the held-out MCC of the action probe, and the cosine similarity between their weight vectors. Each heatmap covers all 80 layers and the final five query tokens. Green outlines in the TruthfulQA cognition panel mark contiguous regions in which MCC ≥ 0.45.

present at many token–layer positions: the green boxes in the TruthfulQA cognition heatmap enclose 28 positions at which MCC ≥ 0.45. Necessity is therefore still meaningfully decodable in the 70B model on TruthfulQA, although less broadly and less strongly than the model's action. Moreover, the lower cognition-probe performance on TruthfulQA than on Arithmetic is consistent with the results for the other models in Section 5.1. Increasing model scale thus does not remove the domain dependence observed in the main experiments: model-adaptive necessity is easier to recover from arithmetic representations than from open-domain factual questions.

Action is consistently separable across layer–token positions. The action probe achieves MCC ≥ 0.4 at 98.75% of positions on Arithmetic and 90.25% on TruthfulQA. The signal spans most layers and token positions rather than being confined to a localized region, indicating that whether the model is about to call a tool is a strongly decodable feature from its residual stream, aligning with the findings in Section 5.2.

Cognition and action become nearly orthogonal at the readout. Although the Arithmetic heatmap contains positive alignment in earlier layers, this alignment does not persist to the position that directly drives generation. At the final token and final layer, the cognition–action cosine similarity is -0.018 on Arithmetic and 0.026 on TruthfulQA. Averaged over the final token in layers 61–80, the corresponding similarities are -0.041 and 0.078. Hence, the larger model reproduces the geometry found in our main results: both signals can be linearly recoverable while their directions are nearly orthogonal at the output-facing representation. These results suggest that the knowing–doing gap is not confined to the smaller models studied in our main experiments.

E Explicitly prompting for verbalized belief of tool-necessity

Due to the limitation of LLMs in verbalizing internal decision processes [17, 9], and the fundamental difference between the task of self-assessment and actual problem solving, in this paper, we followed the approach in recent work that use internal state probing to measure models' cognition of tool-necessity [13, 28]. Nevertheless, for completeness, we also report results obtained using explicit self-assessment prompts.

Specifically, we adopt a two-stage inference procedure. In the first stage, the model is given the same questions from the Arithmetic and TruthfulQA datasets, but instead of solving the problem directly, it is prompted to first decide "whether it is necessary to invoke an external tool" and to answer only with 'yes' or 'no'. In the second stage, the model is instructed to "Now answer the original user request."

Table 6 reports: (1) the MCC between the model's 'yes'/'no' responses and the actual capability-grounded tool necessity defined in Section 4.1; (2) the cognition–execution mismatch rate, defined as the proportion of samples where the model answered 'yes' but did not invoke a tool, or answered 'no'

but eventually invoked one; and (3) the proportion of samples whose eventual tool-calling behavior changed relative to the direct task-oriented prompting setup used in Section 4.2.

The results show that the MCC of explicit 'yes'/'no' judgments is substantially worse at capturing the actual capability-grounded notion of tool necessity. In particular, Llama-3.2-3B-Instruct achieves a negative MCC on TruthfulQA, while Llama-3.1-8B-Instruct simply answers 'no' for every TruthfulQA sample, resulting in an undefined MCC. This behavior implies that the model judges no sample to require a tool, which is clearly inconsistent with the capability measurements reported in Section 4.1. This poor MCC results further show the challenge in distinguishing model-adaptive tool-necessary and tool-unnecessary samples, which is more nuanced than the obvious cases prior work focus on [8, 13, 28].

In contrast, the cognition–execution mismatch rates are noticeably lower than those reported in Section 5.4. Llama-3.1-8B-Instruct even achieves a 0 mismatch rate, meaning that it not only answered 'no' for all samples, but also consistently refrained from making any tool calls. This outcome is somewhat expected: once the 'yes'/'no' response becomes part of the model's context, the model is more likely to remain consistent with that earlier commitment during subsequent generation.

Most importantly, however, Table 6 shows a large "Changed" rate in the third column of each dataset. Relative to direct problem solving with the task-oriented prompts used in our main experiments, explicit self-assessment changes tool-calling behavior on up to nearly 50% of samples. In practical deployments, prompts are typically task-oriented and designed to maximize task performance, rather than to elicit explicit self-assessment. Therefore, this substantial shift in behavior suggests that evaluations based on explicit prompts such as "decide whether it is necessary to invoke an external tool and answer 'yes' or 'no"', as used in some prior work [8, 13], may produce results that diverge significantly from models' actual tool-use behavior under realistic task settings.

Table 6: Tool-call evaluation summary across datasets. For each model and dataset, we report Matthews Correlation Coefficient (MCC), mismatch rate $(\text{yes,false}) + (\text{no,true})$, and the proportion of changed tool-call behavior across variants.

Model	Arithmetic			TruthfulQA		
	MCC	Cog-Exe-Mis. (%)	Changed (%)	MCC	Cog-Exe-Mis. (%)	Changed (%)
Llama-3.1-8B-Instruct	0.0712	5.42	18.20	n/a	0.00	26.93
Llama-3.2-3B-Instruct	0.2251	19.62	29.18	-0.0400	9.67	27.05
Qwen3-4B	0.2330	3.85	29.55	0.1915	36.35	34.64
Qwen3-8B	0.2016	2.12	49.27	0.0538	13.34	20.93

F Computation resources

All experiments were conducted using 2 $\times$ L40s GPUs with CUDA 13.1.

G Limitations

Our experimental setting is relatively restricted to Arithmetic and TruthfulQA with simple calculator and search tools, and extending the analysis to long-horizon, multi-turn agentic environments would be valuable. However, the primary goal of this paper is diagnostic and interpretability-oriented: we aim to isolate the relationship among a model's capability boundary, its internal representation of tool necessity, and its eventual tool-call action. Controlled settings like the ones used allow us to define these quantities cleanly and avoid several confounds that arise in long-horizon tasks.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the findings regarding model-dependent tool necessity, the knowing-doing gap, and the two-stage cognition-execution modeling.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Limitations section is included in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper relies on empirical observations and definitions rather than formal theoretical proofs or theorems.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides details on the datasets used, models evaluated, inference parameters, and the specific formulations for linear probing.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code and data used are publicly released and the link is on the first page of the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All inference hyperparameters are given, and the exact train/test split ratios and the optimizer details for the linear probes are also specified.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The seeds are fixed in probe training for reproducibility, hence there is no statistical randomness in our probing results and analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computation resource used is reported in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses standard LLMs and benign datasets to analyze model behavior, presenting no foreseeable ethical violations.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This paper studies the reliable adaptive tool-calling in LLM agents, which to the best of our knowledge has no negative social impact. Positive impacts include better understanding of LLM tool-calling behaviors, which are trivial enough such that we believe no explicit elaboration is needed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models or sensitive scraped datasets; it analyzes existing open-weights models on standard benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: We cite all the datasets and models but do not highlight their licenses in our paper, as they are widely used research dataset and models.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The newly curated Arithmetic dataset is well-documented in Appendix A, including problem families, proportions, and examples.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The research does not involve human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are the subject of the empirical evaluation and analysis, not a non-standard tool used to generate the core research methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.