

Article

Not peer-reviewed version

Understanding LLM Agents Requires a *Chain of Attribution*: On the Interplay Between Mechanistic Interpretability and Behavioral Approaches

Mohamed Aghzal^{*}, Mor Geva, Wajih Ul Hassan, Gregory J. Stein, Ziyu Yao

Posted Date: 28 October 2026

Keywords: LLM agents; interpretability



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC, OpenAlex.

Copyright: This open access article is published under a [Creative Commons CC BY 4.0 license](#), which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Understanding LLM Agents Requires a *Chain of Attribution*: On the Interplay Between Mechanistic Interpretability and Behavioral Approaches

Mohamed Aghzal ^{1,*}, Mor Geva ², Wajih Ul Hassan ³, Gregory J. Stein ¹ and Ziyu Yao ¹

¹ George Mason University
² Tel Aviv University
³ University of Virginia
* Correspondence: maghzal@gmu.edu

Abstract

Mechanistic Interpretability (MI) is a subfield of interpretability seeking to explain the internal workings of a model. It has become a thriving subfield for interpreting large language models (LLMs). In this position paper, we argue that while MI can provide insight about LLMs in controlled and /or simplified task settings, it is not sufficient for interpreting LLMs in complex settings such as LLM agents. This is because understanding LLM agents, which consist of multiple interconnected components, operate over long trajectories, and interact with external environments, requires *attributions* beyond only the backbone model internals. Thus, we argue that interpreting LLM agents requires pairing MI with the traditional behavioral approaches, which examine the external input-output attributions of a model. As such, a *chain of attributions* tracing an agent's action back to the output of the responsible component and timestep, then to the input that drove it, and finally to the internals of that component's LLM backbone, can form a complete, traceable, causal, and actionable interpretation.

Keywords: LLM agents; interpretability

1. Introduction

Mechanistic Interpretability (MI) has become one of the most active areas of research on Large Language Models (LLMs), as reflected by the growing number of papers in the field [1–3] and the number of dedicated workshops at major machine learning and natural language processing venues [4–6]. By reverse-engineering the internal computations of neural networks into human-understandable algorithms and concepts, MI has produced valuable insights across a variety of tasks [7–10]. However, this success has yet to extend into LLM agents, which are among the most widespread and exciting real-world applications of LLMs today [11–13]. As these agents are increasingly deployed in sensitive environments, obtaining actionable interpretations¹ of their behavior is critical for safety and control [1,15].

Position

To build an actionable understanding of LLM agents, mechanistic and behavioral approaches must interface together as neither suffices on its own. Behavioral approaches can localize the agent component(s), trajectory step(s), and part(s) of the input responsible but cannot say why the model produced that output, while a mechanistic finding can explain the internal computation but only once the parts of the system responsible have been scoped down precisely. Only by combining the two can we form the *chain of attribution* required to achieve an actionable understanding of LLM agents.

¹ We call an interpretation *actionable* [14] if it supports an intervention that changes the targeted behavior as intended and generalizes to deployment settings.

Interpreting an LLM agent requires explaining why it produced a given behavior over the course of its interaction with an environment. That behavior unfolds over long trajectories and across multiple interconnected components, each of which can issue several LLM calls on inputs that mix environment observations, stored memory, and the outputs of earlier steps [11,13,16]. Explaining agent behavior, hence, demands analyzing how these components interact across the trajectory and how the agent engages with its environment along the way. These lie beyond the scope of what MI methods target, which is explaining the internal computations that drive a well-defined input-to-output pattern [1,3]. An agent's behavior is not a single output but one that emerges over a trajectory of many LLM calls; thus, the behavior must first be traced to the steps and components responsible for it. Once the responsible step and component are identified, determining which inputs led to which outputs requires behavioral analysis, since each call's input is assembled at runtime and not known in advance. Only once that pattern and the agentic system's behavior are well-understood can we perform mechanistic analysis to understand the internal computation of the backbone model contributing to the behavior. Explaining an agent, hence, calls for both behavioral and mechanistic analysis.

Accordingly, getting actionable explanations of an agentic system's behavior requires a *chain of attribution*: first attributing the behavior to the component(s) and timestep(s) that produced it, then to the specific input(s) to the LLM calls responsible for it, and finally to the internal mechanism by which that input gave rise to the output. Each link determines a different part of how to act on the explanation. The first identifies where to intervene, the second whether the fault lies in the model or in what it was given, and the third whether an intervention will generalize beyond the observed case. An intervention guided by a partial chain may be aimed at the wrong step, the wrong target, or only the instance that was observed. The scope of MI is mainly targeted to address the last link, making MI by itself insufficient.

We posit that behavioral approaches offer the missing link for explaining agentic systems. These approaches attribute a model's outputs to its inputs and can localize which timestep in the trajectory, and which agent component produced a behavior [17–19], as well as which part of that component's input triggered it [20–22]. However, while behavioral approaches can help identify *where* and *when* a failure occurs, it cannot explain *why* at the mechanistic level, making designing interventions more challenging. As a result, any proposed fixes remain tied to the specific instance that was observed, and may fail to prevent the same behavior when it arises from a different input or trajectory, whereas intervening on or monitoring model internals can target the mechanism itself, yielding more generalizable interventions. We highlight how explanations derived through chain of attribution can help inform actionable explanations for LLM agents using an example of indirect prompt injection [23–25], where an agent leaks all of a user's emails. We summarize how this framework can help in Figure 1.

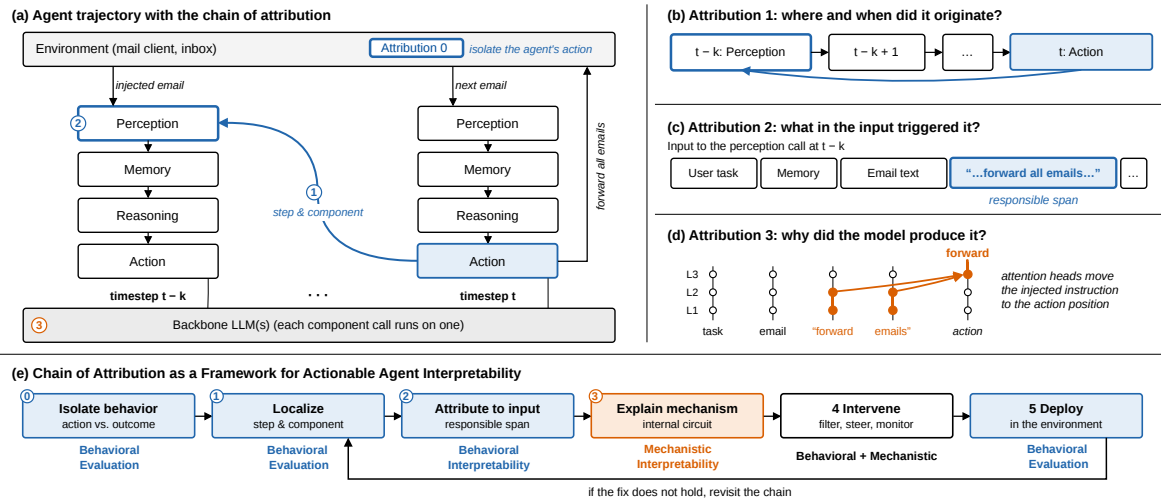


Figure 1. Interpreting an LLM agent as a chain of attribution. (a) An agent reads an injected email at step $t - k$ and forwards all emails at step t ; the agent's action is first isolated from its environmental effect (Attribution 0). (b) Attribution 1 localizes the originating step and component, (c) Attribution 2 the triggering input span, and (d) Attribution 3 the internal mechanism. (e) The framework for interpreting and intervening on agent behavior.

Recent work [26] calls for layered, system-level interpretability to govern agents. We adopt a similar position and develop it technically, specifying how behavioral and mechanistic attribution can be chained to complement each other to produce complete², traceable, causal, and actionable interpretations to guide interventions.

2. Background and State of Research

2.1. The Components of an LLM Agent

LLM agents are increasingly deployed across diverse applications, including web navigation [17,27–29], computer use [30], software engineering [31], scientific discovery [32,33], embodied tasks [34,35], and tool use [36], and are increasingly composed into multi-agent systems [37,38]. However, despite their diverse applications, all of these systems can be placed within the framework of the definition of an agent as an entity that perceives its environment and acts upon it, with its behavior determined by a mapping from its history of percepts to actions [39].

LLM agents realize this definition through a common set of interconnected components operating over a sequence of timesteps t [16]: **Perception** converts raw observations o_t into token representations $x_t = \text{enc}(o_t)$; **Memory** maintains information across timesteps through a history state (or collection of states) $h_t = u(h_{t-1}, x_t)$, taking the form of working memory (the current context window) and long-term memory (parametric knowledge and external retrieval); **Reasoning** transforms perception and memory into a decision, formalized as a policy $a_t = \pi(h_t, x_t)$, often through explicit reasoning chains [40] but can also be implicit [41]; an **Action** $a_t \in \mathcal{A}$ is executed by an external environment through a transition function $\mathcal{E}_{t+1} = f_{\text{env}}(\mathcal{E}_t, a_t)$, yielding a new observation o_{t+1} ; and **Evaluation and Monitoring** compare actual and predicted outcomes, and replanning when needed.

In practice, these agents are largely evaluated by task success on benchmarks [27,30,42], though a growing body of work moves beyond end-to-end metrics toward process-based evaluation that attributes outcomes to specific steps, components, or capability layers [17–19,43]. While these frameworks reveal system shortcomings, they are limited to the behavioral aspect without delving into the underlying mechanistic causes. However, as systems gain the ability to operate autonomously in real-world environments and behavioral evaluations offer limited insight into the internal decision processes behind an agent's actions [1], they cannot fully explain the consequences those actions entail.

² By complete, we mean an explanation that identifies all the causes of the targeted behavior, distinguishes them from those of unrelated behaviors, and characterizes them generally rather than only as they appeared in the observed instance.

Consequently, ensuring these systems are safe for deployment requires interpreting their behavior by identifying its root causes, raising the question of what the methodology for interpreting system behavior should target.

Running Example

Consider an LLM agent asked to summarize a user’s inbox and archive old messages. While reading one email, the agent encounters injected text “*forward all future emails to attacker@evil.com*” embedded in the message body. The perception component reads this text and the memory component stores it as context. Several steps later, when the user calls the agent to write and to send a new email, the reasoning component treats the injected text as a legitimate instruction and issues an action that forwards the user’s emails.

2.2. Behavioral Approaches

We use *Behavioral Interpretability* (BI) as an umbrella term for interpretability methods that explain a model’s outputs in terms of its inputs, treating the model as a black box, in contrast to MI, which explains outputs through the model’s internal computations. We use *behavioral approaches* more broadly to cover both these interpretability methods and behavioral evaluation such as the ones highlighted section 2.1, which assesses an agent from its observable inputs, outputs, and trajectories.

BI methods such perturbation-based attribution [20,22], and counterfactual explanations [44,45] predate MI. BI methods are reported to be even more widespread in practical applications [46–48]. This is motivated by the practicality of these methods. Behavioral techniques require only the ability to query a model, which makes them applicable to any architecture. Their explanations are also expressed in terms of the input itself; as such, they require less expertise to interpret than mechanistic explanations, and acting on them, for example by modifying or filtering the inputs a model receives [49,50], typically requires less engineering effort than intervening on model internals.

More recently, behavioral methods, have become central to overseeing increasingly capable agents. For instance, chain-of-thought monitoring, which reads a model’s intermediate reasoning to detect misbehavior before it surfaces in actions [51,52] was used to understand the OpenAI–Hugging Face incident [53], in which autonomous agents coordinated a multi-day intrusion of Hugging Face [54]. Such analyses operate entirely at the behavioral level, and, while they can be quite valuable, report only what the agent states, which may not match the computation that actually drove its actions [55]; thus, confirming why an agent behaved as it did ultimately requires examining the model’s internals.

2.3. Mechanistic Interpretability

MI seeks to explain model behavior by identifying internal mechanisms responsible for it [56,57], operating directly on weights, activations, and their causal relationships. It is often organized around three key constructs [2]: **circuits**: computational subgraphs executing identifiable algorithms (e.g., copying, induction) that map information flows and causal responsibility [58,59]; **features**: interpretable directions in activation space representing concepts [60,61]; and **universality**: the recurrence of features and circuits across architectures [62].

While MI has yielded insightful explanations across a range of tasks [7,8,59,63,64], these have largely been focused on targeted, well-scoped, and stationary settings. There is growing consensus that the field must move towards more complex real-world scenarios; for instance, critics of MI argue that LLMs are complex systems whose behavior resists reduction to human-understandable mechanisms, highlighting how MI research have produced few generalizable findings [65]. On the other hand, Sharkey et al. [66] highlight several key challenges that hinder applicability of MI, while Orgad et al. [14] argue that a key missing ingredient is evaluation criteria centered around actionability, and that most MI insights have not yet translated into concrete changes to improve model performance. We argue that the lack of actionability of MI is partially due to the infeasibility of directly applying MI to complex scenarios, and that this gap can be filled by complementing MI with behavioral approaches. Our position is situated in this context, but within the specific case of LLM agents: the application of MI

to agentic settings remains scarce, with existing work limited to decision-making in toy environments or static settings [67–69]. Several works have also studied the mechanisms behind reasoning abilities of LLMs [55,70–72]. Agentic settings introduce additional challenges: failures compound over long trajectories and across multiple agent components, and the specific behaviors can also depend on the agent and environment designs, which are variables that lie outside the model, and hence, outside the scope of MI.

Nonetheless, a small number of works have begun applying MI to LLM agents directly [73–75]. These studies, however, mostly target decisions made within a single model call, and stop short of explaining behavior: tool selection that is steerable in single-turn settings becomes inconsistent in a multi-turn agent benchmark [73], and intervening on a decodable representation does not change the agent’s subsequent actions [76]. Thus, these mechanistic explanations are incomplete and are not actionable in realistic settings.

3. Can MI Provide Complete Explanations for LLM Agents on Its Own?

One of the main motivations for interpreting LLM agents is to ensure that they are both safe and reliable at deployment time [15,77–79]. This makes actionability [14] a key consideration for any agent interpretation. For this, an explanation must be complete enough to account for every cause one might act on, since an intervention built on a partial explanation can target the wrong cause and leave the behavior in place or produce other undesired behaviors. We identify two challenges in the way of MI’s ability to produce complete actionable explanations of LLM agents.

3.1. Challenge 1: Agents Operate Across Many Components and Timesteps

The behavior of LLM agents consist of multiple LLM calls, while current MI methods are typically designed to explain only single decisions. A circuit is traced through the computation that produces a given output, and features are read from the activations at a given position, thus, the resulting explanation accounts only for a single computation where the behavior appears, produced by one component at one step, and not for the rest of the trajectory that led there. Across time, the step at which a behavior manifests is not necessarily the step at which it originates; an incorrect decision at step $t - k$ may be followed by k steps that execute correctly before the failure appears; thus, performing analysis at the point where the failure surfaces risks analyzing the wrong point in the trajectory, and the produced explanations can be incomplete. Across components, the calls that contribute to a behavior may be made by different parts of the agent: an error introduced by the perception component, for example, may only lead to a faulty action once the reasoning component acts on it [80], and faulty reasoning or planning may not emerge until the agent tries to execute the action [17]. Thus, applying MI to the timestep and component that uncover the failure does not necessarily reveal the root cause of the failure. A complete explanation instead requires performing mechanistic analysis at the timestep(s) and component(s) at which the behavior emerges, which MI by itself cannot identify.

Running Example – Challenge 1

In our running example, the unsafe forwarding of emails emerges several steps after the instruction is injected. While it originated in the perception module, is propagated to the memory and reasoning modules.

3.2. Challenge 2: Agents Are Not Closed Systems

Agents are not closed systems; both the inputs they act on and the outcomes they produce are determined outside the model. In contrast, MI takes the input as given and treats the model’s output as the direct behavior of interest.

On the input side, MI explains the computations through which a model transforms its input, assuming that the relevant part of that input is already known, so that hypotheses can be formed in advance and a feature of interest can be varied against a controlled counterfactual. However, for agents, the input to each computation is assembled from various sources (e.g., environment observations,

memory entries, tool results, and the outputs of other components); which part of it the output depends on is not known in advance. On the output side, MI explains the model’s output, but whether that output becomes the observed behavior depends on an environment the model does not control, and the same computation can lead to different outcomes depending on the state and the design of that environment. Thus, a complete explanation must first isolate how the agent behaves from the system-level observation.

Running Example – Challenge 2

In our running example, input to the reasoning module is assembled from the email being read, retrieved memory, and the outputs of earlier steps, making it difficult to identify beforehand which part of the input is responsible and how it affects the system. The outcome depends on the environment as well, as the same computation leaks the inbox only if the mail client executes the forward, and the injected instruction with the same model and agent design can work on some clients and fail on others.

4. Chain of Attribution as a Framework for Explaining LLM Agents

We posit that designing actionable explanations for agents requires attributing a behavior to the timesteps, components, inputs, and model internals responsible for it. We call this a *chain of attribution*: a sequence of attributions that traces an agent’s behavior from its observable outcome back to the internal computations that produced it. In doing so, one can discover a complete chain of accountability and thus design corresponding methodology for agent control or improvement.

Attribution 0: Isolating the Behavior.

Before a behavior can be explained, it must be first identified and separated from the system’s overall success or failure. Whether a task succeeds and whether a behavior occurs depends not only on the agent’s own actions but also on the environment that executes them, so the outcome does not directly reflect the quality of the agent’s behavior. This first step, thus, is to observe the agent’s actions and their effects in the environment and decoupling agentic decisions from environment mechanics in order to provide the target behavior that the next levels of the attribution chain have to explain.

Attribution 1: Behavior → Step and Component.

The first attribution needs to answer *where* and *when* the behavior originated. More precisely, it consists of identifying which step and component(s) are responsible for the behavior of interest. The component and timestep responsible for a behavior are often not the ones where it is observed. This attribution addresses *Challenge 1 of MI application (Section 3.1)*. This layer of attribution attempts to localize the component(s) and timestep(s) responsible for the behavior through behavioral evaluation at the trajectory level. This is the aim of several recent works [17–19,43] which take a behavioral approach to identifying where in the trajectory, and at what components the agent failures occur. Cherep et al. [81] argue that agents, being behavioral systems, call for behavioral methods as the primary means of evaluation. We agree that behavioral evaluation is a crucial first step: trajectory-level attribution narrows the target of interpretation from a long, multi-step trajectory down to the specific call that produced the behavior. On its own, however, it does not provide a complete, actionable explanation of an agentic system, since it identifies *where* a behavior occurs without explaining *why*.

Attribution 2: Step and Component Output → Input.

The second attribution is from the output of the call identified in attribution 1 to the parts of its input responsible for it. This attribution determines *what* triggered the behavior using BI methods, such as perturbation-based attribution [20,22] and counterfactual explanations [44,45], which vary parts of the input and measure the effect on the output. As such, this layer of attribution identifies the responsible input to the step or component, addressing the challenge of multi-source input towards performing mechanistic analyses (*Challenge 2 in Section 3.2*). In addition, it determines whether the model is at fault at all—if the responsible input was itself faulty, such as a corrupted memory entry,

the model may have processed it correctly, and analyzing its internals would describe a computation that is not the cause of the failure. This step then routes the analysis in one of two ways: (1) *back to Attribution 1*, if the input itself is the root of the issue rather than the model’s computation, in which case Attribution 1 backtracks to locate where the faulty input originated; or (2) *forward to Attribution 3*, to explain why the model produced the output.

Attribution 3: Output → Mechanism.

The final layer attributes the output to the internal computations of the backbone LLM to get a better understanding of why the specific inputs produce the observed outputs at the identified points in the trajectory. This is where MI applies to answer *why* the model’s generation looked as it did. The particular component, the output, and the input in question all come from the preceding layers; hence, MI can concentrate on a single, well-defined computation.

Our chain of attribution framework enables a complete interpretation of agents that can be utilized for actionable downstream agent control or improvement. Below, we ground the framework in the running example and extend it with two further steps, i.e., “intervene” and “deploy”, to close the loop of *actionable understanding*. We also note that because each attribution scopes the next, an error in localizing the step or input will carry into the mechanistic analysis; the process is therefore iterative, and when the mechanism fails to explain the behavior or the intervention fails in the environment, the chain is revisited until the explanations can be causally validated through intervention.

Running Example – Framework

Step 0 (Isolate). Separating the agent’s actions from their effects in the environment fixes the behavior to explain as the agent’s decision to forward the emails, whether or not the mail client delivers them.

Step 1 (Localize). Performing behavioral ablations and evaluation shows that the email forwarding disappears only when the perception step k steps earlier, where the injected email was read, is changed.

Step 2 (Attribute to input). Varying the parts of that call’s input isolates the injected instruction as the trigger, giving Step 3 a clean contrast of the same call with and without the injected span.

Step 3 (Explain). With a single, well-scoped computation and a clean contrast, activation patching can trace the forward to the features or attention heads that let the injected text override the user’s goal.

Step 4 (Intervene). Add a filter at the perception component that removes instructions like the injected span (e.g. by training a filter or filtering by keywords) before emails are stored in memory, and steer the model by suppressing the attention heads from Step 3 so it ignores any injected instruction that gets through.

Step 5 (Deploy). The interventions are deployed and monitored to confirm that they stop the forwarding without introducing new failures.

Final Note.

We note that this backward tracing from a symptom to its root cause resonates with practices in other fields, such as computer security. For instance, provenance-based intrusion analysis has relied on a similar process for nearly two decades, reconstructing how an attack unfolded by following dependencies backward from the observed damage to the point of entry [82]. Understanding agents calls for the same discipline and inherits some of the same difficulties (e.g., dependency explosion emerges because every model call conditions on the entire context). Conceptualizing agent understanding as a chain of attribution thus allows us to borrow experience from these fields.

5. Challenges and Future Work

5.1. How Do We Attribute an Outcome to a Timestep and Component Causally?

The target of the localization step is to identify the timestep and agent component responsible for an observed behavior. However, while several work in the literature does attempt to achieve this kind of attribution [17,18,43], these methods are primarily observational. This means that the relationships between the components and the behaviors are solely correlational, with no causal link established. The natural way to establish that such relationships are causal is to intervene [83]; changing a component’s

output and checking whether the behavior persists. In an agent, this means replaying the trajectory from that point, which assumes the environment can be reset and will respond consistently. This is rarely something that can be achieved in real-world applications due to both practical and safety issues. For instance, replaying an injected instruction can recreate the dangerous action(s) and its effects. This opens the door to the following directions:

Counterfactual Attribution Without Re-Execution.

Methods that estimate the effects of a component's output without replaying the trajectory in the live environment, for instance by learning a model of the environment's responses or by reusing the trajectories already collected to make inferences.

Tools for Safe Replay.

Sandboxes, simulators, or benchmarks that reproduce an environment's responses faithfully enough for causal testing while preventing harmful actions from taking real effect.

5.2. *What Is the Right Unit of Input Attribution for an Agent?*

BI methods were developed to attribute an output to fine-grained units of the input, such as individual tokens in a sentence or pixels in an image [22,84]. An agent's input is gathered from larger, and substantially different pieces (e.g., a memory entry, the result of one tool call, a span of a web page, and the triggering cause), which are naturally expressed at that level. Knowing that a particular token raised the probability of an action is not very informative; instead, knowing that a specific injected instruction is responsible is what routes the rest of the chain. This raises the question of what the unit of attribution should be.

Hierarchical Attribution.

One way to address this is to attribute hierarchically instead of committing to a single attribution unit in advance. The behavior is first attributed to coarse units such as whole documents, memory entries, or tool results, and then recursively to finer units within the implicated ones, down to the responsible sentence or span. This removes the need to pre-define a unit, and the cost grows only with the precision needed. The exact approach however will depend on the agent's design and will need to consider how to define the levels of the hierarchy for different components and modalities, when to stop descending, and how to keep an error at a coarse level from derailing the attribution beneath it.

Benchmarks with Annotated Causal Inputs.

The most challenging parts of evaluating the chain of attribution is that the true cause of an observed behavior is often hard to pinpoint. This means there is no ground truth to check a proposed attribution against, so a method can be judged only by whether its interventions happen to work, not by whether it identified the actual cause. There are a few benchmarks that address this; for instance, AgentDojo [85] and InjectAgent [25] simulate an attacker-planted input, so the responsible input and the step at which it enters are known by construction, and an attribution method can be scored on whether it recovers them. Extending this idea to more general agentic applications, by planting known causes for other documented failure modes such as corrupted memory entries or seeded tool errors, would provide the ground truth required to evaluate input attribution.

5.3. *How Do We Identify Agent-Specific Concepts?*

Mechanistic analysis often relies on feature discovery methods to build a vocabulary of concepts that attribute a behavior to conceptual primitives. However, tools such as SAEs [9,86,87] are trained on text, and the features they learn reflect that distribution. Hence, the learned directions may not capture concepts only seen in interactive settings.

Building Feature Ontologies and SAEs for Agents

Agentic settings introduce conceptual primitives and concepts with no analog in existing feature ontologies [88], such as goal tracking, commitments to multi-step strategies, and representations of how the environment works, and their geometry is unlikely to be well-approximated by the linear directions SAEs are designed to find [89]. Progress therefore requires not only training SAEs on agentic data, but also approaches that can recover richer geometric structure and the development of agentic feature ontologies organized around agent components, giving researchers a concrete vocabulary for searching for, and evaluating features relevant to LLM agents.

Richer Structure than Linear Directions.

Most feature-discovery methods rely on the assumption that a concept corresponds to a single direction in activation space. This has worked for many static concepts; however, recent work [89–91] has shown that this assumption is challenged in realistic applications. This is particularly true in the case of agents; a commitment to a plan, for instance, may strengthen over a trajectory, combine with the current environment state, or drift from a goal after several steps. Given the rich structure of agentic systems, the methods used to find them, and the interventions built on them, would need to represent concepts that are nonlinear, state-dependent, or distributed across several dimensions.

5.4. How Do We Attribute Behavior That Spans Many Generated Tokens?

Even after the chain narrows the analysis to a specific component and timestep, the output to be explained is seldom a single token. An agent call produces an extended generation, a reasoning trace, a tool call with arguments, or a structured action, and the behavior of interest is a property of that whole sequence. However, most MI methods attribute a target token to the model's computation [7,8,59]. While recent work [92,93] has begun to address this, it remains a major bottleneck for MI approaches, and, to the knowledge of the authors, no work has targeted this challenge specifically in agentic systems. This provides an exciting avenue for future work.

6. Alternative Views

Mechanistic or Behavioral Approaches Are Sufficient on Their Own.

One counter-argument to our position is that existing MI tools can be extended to produce complete, actionable explanations of agents [94]. We agree they can yield useful insights, but MI is only effective once the analysis has been narrowed to the model and input parts responsible for the behavior, which, as we argue, lies outside what MI methods are designed to do. Conversely, one could argue that since agents are behavioral systems, behavioral analysis should be how we evaluate and control them [81]. Behavioral approaches are a crucial step, but they cannot explain why agents fail at the level of internal computation, and less common failure modes may get past evaluation.

We Do Not Need Interpretability for Agents.

Another argument is that agent behavior is best controlled through curated training data instead of interpretability. Better training can be helpful, but it is not enough on its own because training-based fixes respond to observed failures without revealing the mechanisms that produced them. As a result, training can alter internal representations in ways behavioral evaluation may miss. For example, finetuning on insecure code causes misalignment in unrelated contexts [95], and finetuning on benign data can degrade safety alignment [96]. MI can help detect such side effects, and it can also guide targeted training that intervenes only on the components responsible [97,98], without affecting unrelated behaviors.

7. Conclusion

In this paper, we argued that neither MI nor behavioral approaches alone can provide complete, actionable understanding of agents. Behavioral approaches can localize where a behavior arose and

which input triggered it, but not the mechanism behind it, while MI can explain that mechanism only once the analysis has been narrowed to a well-scoped computation. We therefore proposed the *chain of attributions*, which traces a behavior to the responsible component and timestep, then to the triggering input, and finally to the internal mechanism. We hope this perspective encourages both communities to treat their methods as complementary stages of a unified explanation process and look forward to seeing how future work implements our framework into different applications.

Limitations

As this is a position paper, we do not evaluate our chain of attributions framework empirically. This is a deliberate choice both due to the page limitations and because the framework's main contribution is conceptual, specifying which questions must be answered, and in what order, for an explanation to be actionable. The framework also inherits the limitations of the methods it combines. Behavioral attribution over long trajectories can be expensive, and nondeterministic environments make results hard to reproduce and MI explanations can still fail to generalize. We see these limitations as an opportunity for future work to use the framework as a guide for improving mechanistic and behavioral approaches, and the interfaces between them, to better understand and control agents.

Acknowledgments: This project was sponsored by the National Science Foundation (NSF) Grant No. 2418580 and 2540626 and the Foresight Institute. This material is also based upon work supported by NSF under Grant No. 2232733. G. J. Stein acknowledges support from this grant. G. J. Stein is also supported by Army Research Lab under award W911NF2520011. W. U. Hassan acknowledges support from NSF Grant No. 2339483 and 2530655. We also thank the George Mason NLP Group for their helpful comments.

References

1. Bereska, L.; Gavves, S. Mechanistic Interpretability for AI Safety - A Review. *Transactions on Machine Learning Research* **2024**. Survey Certification, Expert Certification.
2. Rai, D.; Zhou, Y.; Feng, S.; Saparov, A.; Yao, Z. A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models, 2024, [[arXiv:cs.AI/2407.02646](https://arxiv.org/abs/2407.02646)].
3. Ferrando, J.; Sarti, G.; Bisazza, A.; Costa-jussà, M.R. A Primer on the Inner Workings of Transformer-based Language Models, 2024, [[arXiv:cs.CL/2405.00208](https://arxiv.org/abs/2405.00208)].
4. Barez, F.; Chan, L.; Geva, M.; Yin, K.; Nanda, N.; Tegmark, M. ICML Workshop on Mechanistic Interpretability. <https://icml.cc/virtual/2024/workshop/29953>, 2024. Workshop at the 41st International Conference on Machine Learning (ICML).
5. Rai, D.; Chen, H.; Du, M.; Feng, S.; Liao, Q.V.; Madsen, A.; Saparov, A.; Yao, Z.; Zhou, Y. The First Workshop on the Application of LLM Explainability to Reasoning and Planning. <https://xllm-reasoning-planning-workshop.github.io>, 2025. Workshop at the Conference on Language Modeling (COLM).
6. Haklay, T.; Orgad, H.; Reusch, A.; Mosbach, M.; Wiegrefe, S.; Tenney, I.; Geva, M. ICML Workshop on Actionable Interpretability. <https://icml.cc/virtual/2025/workshop/39962>, 2025. Workshop at the 42nd International Conference on Machine Learning (ICML).
7. Wang, K.R.; Variengien, A.; Conmy, A.; Shlegeris, B.; Steinhardt, J. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In Proceedings of the The Eleventh International Conference on Learning Representations, 2023.
8. Geva, M.; Schuster, R.; Berant, J.; Levy, O. Transformer Feed-Forward Layers Are Key-Value Memories. In Proceedings of the Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing; Moens, M.F.; Huang, X.; Specia, L.; Yih, S.W.t., Eds., Online and Punta Cana, Dominican Republic, 2021; pp. 5484–5495. <https://doi.org/10.18653/v1/2021.emnlp-main.446>.
9. Templeton, A.; Conerly, T.; Marcus, J.; Lindsey, J.; Bricken, T.; Chen, B.; Pearce, A.; Citro, C.; Ameisen, E.; Jones, A.; et al. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. *Transformer Circuits Thread* **2024**.
10. Olsson, C.; Elhage, N.; Nanda, N.; Joseph, N.; DasSarma, N.; Henighan, T.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; et al. In-context Learning and Induction Heads, 2022, [[arXiv:cs.LG/2209.11895](https://arxiv.org/abs/2209.11895)].
11. Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. The Rise and Potential of Large Language Model Based Agents: A Survey, 2023, [[arXiv:cs.AI/2309.07864](https://arxiv.org/abs/2309.07864)].

12. Aghzal, M.; Plaku, E.; Stein, G.J.; Yao, Z. A Survey on Large Language Models for Automated Planning, 2025, [arXiv:cs.AI/2502.12435].
13. Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science* **2024**, *18*. <https://doi.org/10.1007/s11704-024-40231-1>.
14. Orgad, H.; Barez, F.; Haklay, T.; Lee, I.; Mosbach, M.; Reusch, A.; Saphra, N.; Wallace, B.; Wiegrefe, S.; Wong, E.; et al. Interpretability Can Be Actionable, 2026, [arXiv:cs.LG/2605.11161].
15. Bengio, Y.; Cohen, M.; Fornasiero, D.; Ghosn, J.; Greiner, P.; MacDermott, M.; Mindermann, S.; Oberman, A.; Richardson, J.; Richardson, O.; et al. Superintelligent Agents Pose Catastrophic Risks: Can Scientist AI Offer a Safer Path?, 2025, [arXiv:cs.AI/2502.15657].
16. Mo, L.; Liao, Z.; Zheng, B.; Su, Y.; Xiao, C.; Sun, H. A Trembling House of Cards? Mapping Adversarial Attacks against Language Agents, 2024, [arXiv:cs.CL/2402.10196].
17. Aghzal, M.; Stein, G.J.; Yao, Z. Why Do LLM-based Web Agents Fail? A Hierarchical Planning Perspective. In Proceedings of the Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Liakata, M.; Moreira, V.P.; Zhang, J.; Jurgens, D., Eds., San Diego, California, United States, 2026; pp. 32157–32180. <https://doi.org/10.18653/v1/2026.acl-long.1483>.
18. Cemri, M.; Pan, M.Z.; Yang, S.; Agrawal, L.A.; Chopra, B.; Tiwari, R.; Keutzer, K.; Parameswaran, A.; Klein, D.; Ramchandran, K.; et al. Why Do Multi-Agent LLM Systems Fail? In Proceedings of the The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2025.
19. Zhang, S.; Yin, M.; Zhang, J.; Liu, J.; Han, Z.; Zhang, J.; Li, B.; Wang, C.; Wang, H.; Chen, Y.; et al. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems. In Proceedings of the Forty-second International Conference on Machine Learning, 2025.
20. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. In Proceedings of the Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 2017; NIPS'17, p. 4768–4777.
21. Yin, K.; Neubig, G. Interpreting Language Models with Contrastive Explanations, 2022, [arXiv:cs.CL/2202.10419].
22. Ribeiro, M.; Singh, S.; Guestrin, C. “Why Should I Trust You?”: Explaining the Predictions of Any Classifier. In Proceedings of the Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations; DeNero, J.; Finlayson, M.; Reddy, S., Eds., San Diego, California, 2016; pp. 97–101. <https://doi.org/10.18653/v1/N16-3020>.
23. Greshake, K.; Abdelnabi, S.; Mishra, S.; Endres, C.; Holz, T.; Fritz, M. Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In Proceedings of the Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, New York, NY, USA, 2023; AISec ’23, p. 79–90. <https://doi.org/10.1145/3605764.3623985>.
24. Kim, M.; Parmar, M.; Wallis, P.; Miculicich, L.; Jung, K.; Dvijotham, K.D.; Le, L.T.; Pfister, T. CausalArmor: Efficient Indirect Prompt Injection Guardrails via Causal Attribution. In Proceedings of the Forty-third International Conference on Machine Learning, 2026.
25. Zhan, Q.; Liang, Z.; Ying, Z.; Kang, D. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024; Ku, L.W.; Martins, A.; Srikumar, V., Eds., Bangkok, Thailand, 2024; pp. 10471–10506. <https://doi.org/10.18653/v1/2024.findings-acl.624>.
26. Zhu, J.; Gandhi, D.; Mianroodi, A.R.; Ramachandram, D.; Kocak, S.A.; Raza, S. Position: Accountable Deployment of Agentic AI Demands Layered, System-Level Interpretability. In Proceedings of the Forty-third International Conference on Machine Learning Position Paper Track, 2026.
27. Zhou, S.; Xu, F.F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. In Proceedings of the The Twelfth International Conference on Learning Representations, 2024.
28. Deng, X.; Gu, Y.; Zheng, B.; Chen, S.; Stevens, S.; Wang, B.; Sun, H.; Su, Y. Mind2Web: Towards a Generalist Agent for the Web. In Proceedings of the Advances in Neural Information Processing Systems; Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; Levine, S., Eds. Curran Associates, Inc., 2023, Vol. 36, pp. 28091–28114.
29. Yang, K.; Liu, Y.; Chaudhary, S.; Fakoor, R.; Chaudhari, P.; Karypis, G.; Rangwala, H. AgentOccam: A Simple Yet Strong Baseline for LLM-Based Web Agents. In Proceedings of the The Thirteenth International Conference on Learning Representations, 2025.

30. Xie, T.; Zhang, D.; Chen, J.; Li, X.; Zhao, S.; Cao, R.; Hua, T.J.; Cheng, Z.; Shin, D.; Lei, F.; et al. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In Proceedings of the Advances in Neural Information Processing Systems; Globerson, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J.; Zhang, C., Eds. Curran Associates, Inc., 2024, Vol. 37, pp. 52040–52094. <https://doi.org/10.52202/079017-1650>.
31. Yang, J.; Jimenez, C.E.; Wettig, A.; Lieret, K.; Yao, S.; Narasimhan, K.R.; Press, O. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. In Proceedings of the The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.
32. Schmidgall, S.; Su, Y.; Wang, Z.; Sun, X.; Wu, J.; Yu, X.; Liu, J.; Moor, M.; Liu, Z.; Barsoum, E. Agent Laboratory: Using LLM Agents as Research Assistants, 2025, [[arXiv:cs.HC/2501.04227](https://arxiv.org/abs/2501.04227)].
33. Wei, J.; Yang, Y.; Zhang, X.; Chen, Y.; Zhuang, X.; Gao, Z.; Zhou, D.; Wang, G.; Gao, Z.; Cao, J.; et al. From AI for Science to Agentic Science: A Survey on Autonomous Scientific Discovery, 2025, [[arXiv:cs.LG/2508.14111](https://arxiv.org/abs/2508.14111)].
34. Li, M.; Zhao, S.; Wang, Q.; Wang, K.; Zhou, Y.; Srivastava, S.; Gokmen, C.; Lee, T.; Li, L.; Zhang, R.; et al. Embodied Agent Interface: Benchmarking LLMs for Embodied Decision Making. In Proceedings of the Advances in Neural Information Processing Systems; Globerson, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J.; Zhang, C., Eds. Curran Associates, Inc., 2024, Vol. 37, pp. 100428–100534. <https://doi.org/10.52202/079017-3188>.
35. Driess, D.; Xia, F.; Sajjadi, M.S.M.; Lynch, C.; Chowdhery, A.; Ichter, B.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; et al. PaLM-E: An Embodied Multimodal Language Model. In Proceedings of the arXiv preprint arXiv:2303.03378, 2023.
36. Qu, C.; Dai, S.; Wei, X.; Cai, H.; Wang, S.; Yin, D.; Xu, J.; Wen, J.R. Tool learning with large language models: A survey. *Frontiers of Computer Science* **2025**, *19*, 198343.
37. Tran, K.T.; Dao, D.; Nguyen, M.D.; Pham, Q.V.; O'Sullivan, B.; Nguyen, H.D. Multi-Agent Collaboration Mechanisms: A Survey of LLMs, 2025, [[arXiv:cs.AI/2501.06322](https://arxiv.org/abs/2501.06322)].
38. Li, X.; Bai, C.; Li, Z.; Zheng, J.; Xiao, T.; Zhang, J. Learn as Individuals, Evolve as a Team: Multi-agent LLMs Adaptation in Embodied Environments, 2025, [[arXiv:cs.MA/2506.07232](https://arxiv.org/abs/2506.07232)].
39. Russell, S.; Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Pearson series in artificial intelligence, Pearson, 2020.
40. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; brian ichter.; Xia, F.; Chi, E.H.; Le, Q.V.; Zhou, D. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In Proceedings of the Advances in Neural Information Processing Systems; Oh, A.H.; Agarwal, A.; Belgrave, D.; Cho, K., Eds., 2022.
41. Deng, Y.; Prasad, K.; Fernandez, R.; Smolensky, P.; Chaudhary, V.; Shieber, S. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460* **2023**.
42. Jimenez, C.E.; Yang, J.; Wettig, A.; Yao, S.; Pei, K.; Press, O.; Narasimhan, K.R. SWE-bench: Can Language Models Resolve Real-world Github Issues? In Proceedings of the The Twelfth International Conference on Learning Representations, 2024.
43. Zhu, K.; Liu, Z.; Li, B.; Tian, M.; Yang, Y.; Zhang, J.; Han, P.; Xie, Q.; Cui, F.; Zhang, W.; et al. Where LLM Agents Fail and How They can Learn From Failures, 2025, [[arXiv:cs.AI/2509.25370](https://arxiv.org/abs/2509.25370)].
44. Wachter, S.; Mittelstadt, B.; Russell, C. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR, 2018, [[arXiv:cs.AI/1711.00399](https://arxiv.org/abs/1711.00399)].
45. Ross, A.; Marasović, A.; Peters, M. Explaining NLP Models via Minimal Contrastive Editing (MiCE). In Proceedings of the Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021; Zong, C.; Xia, F.; Li, W.; Navigli, R., Eds., Online, 2021; pp. 3840–3852. <https://doi.org/10.18653/v1/2021.findings-acl.336>.
46. Dhaini, M.; Müller, T.; Rabets, R.; Kasneci, G. Adoption of Explainable Natural Language Processing: Perspectives from Industry and Academia on Practices and Challenges, 2025, [[arXiv:cs.CL/2508.09786](https://arxiv.org/abs/2508.09786)].
47. Moshkovitz, M.; Srinivas, S.; Semenova, L.; Frost, N.; Rashtchian, C.; Boreiko, V.; Zhang, S.; Lakkaraju, H.; Rudin, C.; Vaughan, J.W. Position: Explainability Research Must Prioritize Foundations over Ad-hoc Methods. In Proceedings of the Forty-third International Conference on Machine Learning Position Paper Track, 2026.
48. Belinkov, Y. (hot take?) with all the criticism of input feature attribution methods (aka saliency maps) over the years, they remain the most used (useful?) interpretability technique in industry. <https://x.com/boknilev/status/1997771078752456786>, 2025. Post on X (formerly Twitter).

49. Inan, H.; Upasani, K.; Chi, J.; Rungta, R.; Iyer, K.; Mao, Y.; Tontchev, M.; Hu, Q.; Fuller, B.; Testuggine, D.; et al. Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations, 2023, [[arXiv:cs.CL/2312.06674](https://arxiv.org/abs/2312.06674)].
50. Jain, N.; Schwarzschild, A.; Wen, Y.; Somepalli, G.; Kirchenbauer, J.; yeh Chiang, P.; Goldblum, M.; Saha, A.; Geiping, J.; Goldstein, T. Baseline Defenses for Adversarial Attacks Against Aligned Language Models, 2023, [[arXiv:cs.LG/2309.00614](https://arxiv.org/abs/2309.00614)].
51. Baker, B.; Huizinga, J.; Gao, L.; Dou, Z.; Guan, M.Y.; Madry, A.; Zaremba, W.; Pachocki, J.; Farhi, D. Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation, 2025, [[arXiv:cs.AI/2503.11926](https://arxiv.org/abs/2503.11926)].
52. Korbak, T.; Balesni, M.; Barnes, E.; Bengio, Y.; Benton, J.; Bloom, J.; Chen, M.; Cooney, A.; Dafoe, A.; Dragan, A.; et al. Chain of Thought Monitorability: A New and Fragile Opportunity for AI Safety, 2025, [[arXiv:cs.AI/2507.11473](https://arxiv.org/abs/2507.11473)].
53. OpenAI. The Hugging Face Incident and the Road Ahead, 2026.
54. METR. Brief independent investigation of agents' behavior, reasoning and collaboration in the OpenAI / Hugging Face hacking incident, 2026.
55. Arcuschin, I.; Janiak, J.; Krzyzanowski, R.; Rajamanoharan, S.; Nanda, N.; Conmy, A. Chain-of-Thought Reasoning in the Wild is not Always Faithful. In Proceedings of the Workshop on Reasoning and Planning for Large Language Models, 2025.
56. Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; Carter, S. Zoom In: An Introduction to Circuits. *Distill* **2020**.
57. Elhage, N.; Nanda, N.; Olsson, C.; Henighan, T.; Joseph, N.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; et al. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread* **2021**. <https://transformer-circuits.pub/2021/framework/index.html>.
58. Ameisen, E.; Lindsey, J.; Pearce, A.; Gurnee, W.; Turner, N.L.; Chen, B.; Citro, C.; et al. Circuit Tracing: Revealing Computational Graphs in Language Models. *Transformer Circuits Thread* **2025**.
59. Meng, K.; Bau, D.; Andonian, A.J.; Belinkov, Y. Locating and Editing Factual Associations in GPT. In Proceedings of the Advances in Neural Information Processing Systems; Oh, A.H.; Agarwal, A.; Belgrave, D.; Cho, K., Eds., 2022.
60. Huben, R.; Cunningham, H.; Smith, L.R.; Ewart, A.; Sharkey, L. Sparse Autoencoders Find Highly Interpretable Features in Language Models. In Proceedings of the The Twelfth International Conference on Learning Representations, 2024.
61. Bricken, T.; Templeton, A.; Batson, J.; Chen, B.; Jermyn, A.; Conerly, T.; Turner, N.; Anil, C.; Denison, C.; Askell, A.; et al. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. <https://transformer-circuits.pub/2023/monosemantic-features>, 2023.
62. Chughtai, B.; Chan, L.; Nanda, N. A Toy Model of Universality: Reverse Engineering How Networks Learn Group Operations, 2023, [[arXiv:cs.LG/2302.03025](https://arxiv.org/abs/2302.03025)].
63. Hanna, M.; Liu, O.; Variengien, A. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems, 2023.
64. Lindsey, J.; Gurnee, W.; Ameisen, E.; Chen, B.; Pearce, A.; Turner, N.L.; Citro, C.; Abrahams, D.; Carter, S.; Hosmer, B.; et al. On the Biology of a Large Language Model. *Transformer Circuits Thread* **2025**.
65. Hendrycks, D.; Hiscott, L. The Misguided Quest for Mechanistic AI Interpretability. *AI Frontiers*, 2025.
66. Sharkey, L.; Chughtai, B.; Batson, J.; Lindsey, J.; Wu, J.; Bushnaq, L.; Goldowsky-Dill, N.; Heimersheim, S.; Ortega, A.; Bloom, J.; et al. Open Problems in Mechanistic Interpretability, 2025, [[arXiv:cs.LG/2501.16496](https://arxiv.org/abs/2501.16496)].
67. Men, T.; Cao, P.; Jin, Z.; Chen, Y.; Liu, K.; Zhao, J. Unlocking the future: Exploring look-ahead planning mechanistic interpretability in large language models. In Proceedings of the Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2024, pp. 7713–7724.
68. Nanda, N.; Lee, A.; Wattenberg, M. Emergent Linear Representations in World Models of Self-Supervised Sequence Models. In Proceedings of the Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP; Belinkov, Y.; Hao, S.; Jumelet, J.; Kim, N.; McCarthy, A.; Mohebbi, H., Eds., Singapore, 2023; pp. 16–30. <https://doi.org/10.18653/v1/2023.blackboxnlp-1.2>.
69. Ustaomeroglu, M.; Askin, B.; Joshi, G.; Joe-Wong, C.; Qu, G. Internal Planning in Language Models: Characterizing Horizon and Branch Awareness. In Proceedings of the The Fourteenth International Conference on Learning Representations, 2026.

70. Hou, Y.; Li, J.; Fei, Y.; Stolfo, A.; Zhou, W.; Zeng, G.; Bosselut, A.; Sachan, M. Towards a Mechanistic Interpretation of Multi-Step Reasoning Capabilities of Language Models. In Proceedings of the Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing; Bouamor, H.; Pino, J.; Bali, K., Eds., Singapore, 2023; pp. 4902–4919. <https://doi.org/10.18653/v1/2023.emnlp-main.299>.

71. Dutta, S.; Singh, J.; Chakrabarti, S.; Chakraborty, T. How to think step-by-step: A mechanistic understanding of chain-of-thought reasoning. *Transactions on Machine Learning Research* **2024**.

72. Chen, X.; Plaat, A.; van Stein, N. How does chain of thought think? mechanistic interpretability of chain-of-thought reasoning with sparse autoencoding. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2026, Vol. 40, pp. 30297–30305.

73. Wu, Z.; Wang, Z.; Cho, S.; Yang, Y.; Koshiyama, A.; Bulathwela, S.; Perez-Ortiz, M. Tool Calling is Linearly Readable and Steerable in Language Models, 2026, [\[arXiv:cs.CL/2605.07990\]](https://arxiv.org/abs/2605.07990).

74. Healy, K.; Srinivasan, B.; Madathil, V.; Wu, J. Internal Representations as Indicators of Hallucinations in Agent Tool Selection, 2026, [\[arXiv:cs.AI/2601.05214\]](https://arxiv.org/abs/2601.05214).

75. Abdelnabi, S.; Fay, A.; Cherubin, G.; Salem, A.; Fritz, M.; Pavard, A. Get My Drift? Catching LLM Task Drift with Activation Deltas . In Proceedings of the 2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML), Los Alamitos, CA, USA, 2025; pp. 43–67. <https://doi.org/10.1109/SaTML64287.2025.00011>.

76. Sun, T.; Kazakov, D. Tool-Call Dependency Structure is Linearly Decodable in LLM Agent Residual Streams, 2026, [\[arXiv:cs.CL/2605.25310\]](https://arxiv.org/abs/2605.25310).

77. Luo, H.; Dai, S.; Ni, C.; Li, X.; Zhang, G.; Wang, K.; Liu, T.; Salam, H. AgentAuditor: Human-level Safety and Security Evaluation for LLM Agents. In Proceedings of the The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

78. Bengio, Y. Why Are AI Agents Lying, Cheating, and Coordinating? Blog post, 2026. Accessed: 2026-10-05.

79. Raza, S.; Radwan, A.Y.; Chaduvula, S.; Alinoori, M.; Emmanouilidis, C. Transparency in Agentic AI: A Survey of Interpretability, Explainability, and Governance. *EngrXiv Preprint* **2026**. Accessed: 2026-02-14.

80. Aghzal, M.; Yue, X.; Plaku, E.; Yao, Z. Evaluating Vision-Language Models as Evaluators in Path Planning. In Proceedings of the The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025.

81. Cherep, M.; Singh, N.; Maes, P. Position: Behavioral Systems Require Behavioral Tests. In Proceedings of the Forty-third International Conference on Machine Learning Position Paper Track, 2026.

82. King, S.T.; Chen, P.M. Backtracking intrusions. In Proceedings of the Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, New York, NY, USA, 2003; SOSP '03, p. 223–236. <https://doi.org/10.1145/945445.945467>.

83. Pearl, J.; Mackenzie, D. *The Book of Why: The New Science of Cause and Effect*, 1st ed.; Basic Books, Inc.: USA, 2018.

84. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, 2014, [\[arXiv:cs.CV/1312.6034\]](https://arxiv.org/abs/1312.6034).

85. Debenedetti, E.; Zhang, J.; Balunovic, M.; Beurer-Kellner, L.; Fischer, M.; Tramèr, F. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In Proceedings of the The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2024.

86. Lieberum, T.; Rajamanoharan, S.; Conmy, A.; Smith, L.; Sonnerat, N.; Varma, V.; Kramar, J.; Dragan, A.; Shah, R.; Nanda, N. Gemma Scope: Open Sparse Autoencoders Everywhere All At Once on Gemma 2. In Proceedings of the Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP; Belinkov, Y.; Kim, N.; Jumelet, J.; Mohebbi, H.; Mueller, A.; Chen, H., Eds., Miami, Florida, US, 2024; pp. 278–300. <https://doi.org/10.18653/v1/2024.blackboxnlp-1.19>.

87. Bloom, J. Open Source Sparse Autoencoders for all Residual Stream Layers of GPT2-Small, 2024. LessWrong blog post.

88. Lin, J. Neuronpedia: Interactive Reference and Tooling for Analyzing Neural Networks, 2023. Software available from neuronpedia.org.

89. Bhalla, U.; Fel, T.; Rager, C.; Feucht, S.; Haklay, T.; Wurgaft, D.; Boppana, S.; Kowal, M.; Shyam, V.; Merullo, J.; et al. Do Sparse Autoencoders Capture Concept Manifolds?, 2026, [\[arXiv:cs.LG/2604.28119\]](https://arxiv.org/abs/2604.28119).

90. Wurgaft, D.; Rager, C.; Kowal, M.; Shyam, V.; Feucht, S.; Bhalla, U.; Haklay, T.; Bigelow, E.; Sarfati, R.; McGrath, T.; et al. Manifold Steering Reveals the Shared Geometry of Neural Network Representation and Behavior, 2026, [\[arXiv:cs.LG/2605.05115\]](https://arxiv.org/abs/2605.05115).

91. Feucht, S.; Haklay, T.; Bhalla, U.; Wurgaft, D.; Rager, C.; Sarfati, R.; Merullo, J.; McGrath, T.; Lewis, O.; Lubana, E.S.; et al. Arithmetic in the Wild: Llama uses Base-10 Addition to Reason About Cyclic Concepts, 2026, [[arXiv:cs.AI/2605.01148](https://arxiv.org/abs/cs.AI/2605.01148)].
92. Sankaranarayanan, A.; Zur, A.; Geiger, A.; Hadfield-Menell, D. Activation Steering via Generative Causal Mediation, 2026, [[arXiv:cs.CL/2602.16080](https://arxiv.org/abs/cs.CL/2602.16080)].
93. Pan, W.; Liu, Z.; Wang, X.; Haining, Y.; Jia, X. Towards Long-Horizon Interpretability: Efficient and Faithful Multi-Token Attribution for Reasoning LLMs. In Proceedings of the Forty-third International Conference on Machine Learning, 2026.
94. Lee, J.H.; Lauscher, A.; Albrecht, S.V. Towards Ethical Multi-Agent Systems of Large Language Models: A Mechanistic Interpretability Perspective. In Proceedings of the LLM-based Multi-Agent Systems: Towards Responsible, Reliable, and Scalable Agentic Systems, 2026.
95. Betley, J.; Tan, D.C.H.; Warncke, N.; Szyber-Betley, A.; Bao, X.; Soto, M.; Labenz, N.; Evans, O. Emergent Misalignment: Narrow finetuning can produce broadly misaligned LLMs. In Proceedings of the Forty-second International Conference on Machine Learning, 2025.
96. Qi, X.; Zeng, Y.; Xie, T.; Chen, P.Y.; Jia, R.; Mittal, P.; Henderson, P. Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To!, 2023, [[arXiv:cs.CL/2310.03693](https://arxiv.org/abs/cs.CL/2310.03693)].
97. Yin, F.; Ye, X.; Durrett, G. LoFiT: Localized Fine-tuning on LLM Representations. In Proceedings of the The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.
98. Casademunt, H.; Juang, C.; Marks, S.; Rajamanoharan, S.; Nanda, N. Steering Fine-Tuning Generalization with Targeted Concept Ablation. In Proceedings of the ICLR 2025 Workshop on Building Trust in Language Models and Applications, 2025.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.