Journal of COMPUTER AND FORENSIC SCIENCES

CFS 2026, Vol. 5, Issue 2
https://doi.org/10.5937/jcfs5-68363
Original research paper

Derivation Boundaries: A Structural Source of Faithfulness Failure in Large Language Models

Vaishnavi Sreekumar[1]

[1] Student at SRH Hochschule Heidelberg, Germany

Corresponding author: vaishnavisreekumar28@gmail.com

Received: June 17, 2026 • Accepted: July 16, 2026 • Published: September 11, 2026

ONLINE FIRST

Abstract: Faithfulness to supplied information is not uniform in large language models: some rules, thresholds, or features shape outputs correctly while others do not, with nothing in a model's own explanations distinguishing the two. An industrial classification task motivates this observation, where supplying a classifier's exact decision thresholds improved accuracy while degrading minority-class recall, raising the question of where such failures occur. Because that setting involved a closed model and proprietary data, the same question is examined in two domains: airline baggage fee and tax calculation. Across domains, models reliably handle directly available values but fail when a known fact must be transformed into a new value through a rule-conditioned lookup, points named here as derivation boundaries. Two error patterns recur there: misassignment, in which a valid value is assigned to the wrong side of a rule boundary, and fabrication, in which a value unsupported by the rules is introduced and justified as though retrieved correctly. Layer-wise analysis using the logit lens suggests a mechanistic explanation. Directly available values stabilize early, whereas boundary-dependent values commit later and less stably, with correct alternatives often remaining competitive until the final layers. Faithfulness failures concentrate at derivation boundaries rather than spreading uniformly, a pattern layer-wise analysis links to delayed internal resolution.

Keywords: large language models; chain-of-thought faithfulness; rule-guided reasoning; derivation boundary; hallucination.

1. INTRODUCTION

Large language models are increasingly deployed as components in pipelines where they are given external information to support a decision, like retrieved documents, computed features, domain-specific rules, threshold values, or other pre-processed quantities, on the implicit expectation that having this information is sufficient to use it correctly. Yet, having access to such information does not affect a model's output uniformly. Some of the information that is given to the model within a single response is incorporated faithfully

Citation: V. Sreekumar, Derivation Boundaries: A Structural Source of Faithfulness Failure in Large Language Models," *Journal of Computer and Forensic Sciences*, vol. 5, no. 2, pp. , 2026, https://doi.org/10.5937/jcfs5-68363. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) 4.0 International License (https://creativecommons.org/licenses/by/4.0/).

into its stated reasoning and final answer while some is not, and nothing in the model's own account of either case marks the difference between them, a pattern Turpin et al. demonstrate directly by showing that chain-of-thought explanations can systematically omit a factor that demonstrably influenced the model's answer [1].

This investigation began in my master's thesis [10], which examined an industrial time-series classification problem, distinguishing two types of events in a manufacturing process from sensor sequences. There, a large language model was given the exact decision thresholds used by a trained classifier with the aim of using the model's verbalized reasoning as an interpretable substitute for the classifier itself. The thresholds were stated explicitly in the same numeric form and units the classifier used internally, so the model was missing nothing it needed in principle. Supplying them raised overall classification accuracy while collapsing recall on the minority class, which is the harder and more consequential class. This is not the result a faithfully rule-following model should produce. A threshold handed to a model outright should not make its performance worse.

This anomaly motivates the central question of present work: where do faithfulness failures occur, and why do they occur there rather than being distributed uniformly across a model's reasoning process?

The failures are not diffuse. Instead, they concentrate on a specific class of computational step, one at which a directly available fact must be converted into a further value through a rule-conditioned operation rather than by direct restatement or elementary computation. This step recurs across every domain examined and can be identified from a task's structure before a model is ever run. Using layer-wise analysis, this paper shows that the values this step requires are committed to later and less stably than directly available values inside the model, which accounts for why it is especially vulnerable to failure.

ONLINE FIRST

2. EXPERIMENTAL SETUP

A setting that tests this claim requires every intermediate value to be verifiable, the model's weights and decoding behaviour to be fully inspectable and reproducible, and a task different enough from industrial sensor classification that a recurrence of the same pattern cannot be attributed to some shared peculiarity of the original data. The thesis observation that motivates this paper meets none of these conditions on its own, as it is a closed model, a proprietary dataset, and a single task that cannot be varied or inspected further. What follows traces the same question through three settings, an industrial classifier's threshold, an airline's fee table, and a tax computation, each chosen to test a different part of this requirement, before turning to what the three settings turn out to share.

Beginning with industrial classification, not all the thresholds supplied to the model in this task were of the same kind, though nothing marked this distinction in how they were presented. Some depended on values already sitting in the input, like a raw count or a directly logged sensor reading that is available without further computation or look-ups. Others depended on a look-up or a quantity that had to be computed from the raw sequence. This computed value then had to be checked against a table of class-specific patterns to determine which threshold applied. The model applied the first kind of threshold correctly and consistently. The second kind was handled differently. The threshold that

mattered most for distinguishing the minority class had been given to the model explicitly. But it never performed the lookup that the threshold depended on, which meant checking the computed ratio against the table of class-specific patterns. Nothing in its reasoning trace admitted that this lookup had been skipped. This raises a question a single observation cannot answer. If a model handles most of what it is given correctly and fails on one specific case, is the failure a peculiarity of that one threshold, that one dataset, that one model, or is something more general being exposed by it?

A public benchmark of rule-guided computation problems, RuleArena [3], structurally unrelated to industrial sensor classification, offers a way to test this. In a task requiring a model to compute the total cost of a passenger's checked baggage, an itinerary, a set of bags, and a fee policy are given along with each bag's stated dimensions and weight. Summing a bag's measurements into a total and comparing that total or the bag's stated weight against a stated threshold is handled correctly in most cases. The fee a bracketed policy table assigns once a bag is determined to be oversize or overweight is a different matter. Once a bag's weight places it in a specific bracket of the policy's table, its fee should follow directly from that bracket. In practice, the fee reported does not always match the bracket the bag's own stated weight falls into. The arithmetic that precedes this step is reliable. The lookup that follows it is not.

A second domain from the same benchmark, computing the amount owed or refunded on a partially completed tax form, shows the same shape of failure in a setting that shares nothing with baggage fees beyond its underlying structure. Income subtotals, summed from stated components across several lines of the form, present no difficulty. The standard deduction and the tax owed, each of which depends on a lookup conditioned on a stated fact rather than on summation, present the same difficulty seen elsewhere. At the line requiring a tax-table lookup, the model sometimes states that the table is not provided in the information given and supplies a specific figure regardless, introduced with the same confidence used for every value it has genuinely computed.

An industrial classifier's decision threshold, an airline's fee bracket, and a tax table share no vocabulary, no numeric range, and no subject matter. Set beside one another, the three failures share something else. In each case a quantity is available as a directly stated fact. But the value a governing rule requires is not that quantity itself. It is a further value reachable only through a lookup conditioned on it, like a row in a table, a bracket on a schedule, or a ratio's threshold. The fact is present, but the lookup is not performed. A plausible value takes its place, carried forward with no change in register from a value that was genuinely retrieved.

There is a point in a reasoning chain at which a directly available fact gives way to this kind of requirement. That point is what this paper calls a derivation boundary. Three failures that shared nothing else had already crossed it (Figure 1).

ONLINE FIRST

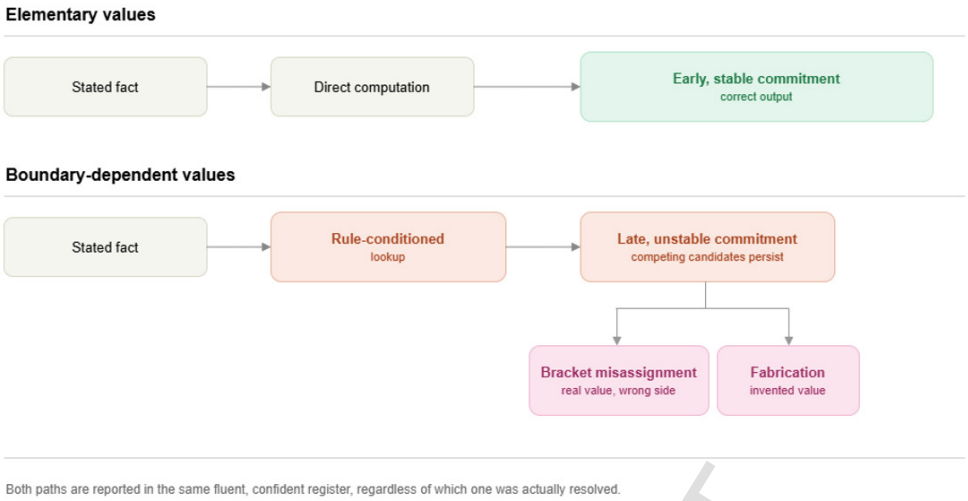


Figure 1. *Overview of the paper's argument: elementary values commit early and stably; boundary-dependent values commit late and unstably, producing either misassignment or fabrication.*

ONLINE FIRST

3. DERIVATION BOUNDARIES

A value required by a task can relate to the information given to a model in one of two ways. It can be directly available – stated outright, or obtainable from stated values by an operation simple enough that producing it and producing fluent text about it are close to the same act as a sum, a restatement, a direct comparison against a quantity already in hand. Or it can be boundary-dependent, which means its determination requires a further operation conditioned on a stated fact like a value's bracket on a schedule, a row in a table indexed by a stated quantity, a category assigned by a threshold defined over something not itself given. The first kind of value can be produced without resolving anything beyond what is already present in the input, but the second cannot.

A derivation boundary, formally, is the step in a reasoning chain at which the first kind of value gives way to a requirement for the second, which is the point at which a fact already in hand must be converted through a rule-conditioned lookup or comparison into a value the input does not itself contain. This is a property of a task's structure, fixed by its rules and its given information and assessable without reference to any model's output. Whether a given step is elementary or boundary-dependent can be settled by inspecting the rule that step depends on, even before that rule is ever handed to a model (Figure 1).

This distinction yields a hypothesis precise enough to be wrong. If a derivation boundary marks a genuine point of failure, rather than a label applied retroactively to wherever a model happens to err, failures should concentrate at exactly these steps regardless of domain and should be comparatively absent from the elementary steps surrounding them. The sections that follow test this directly.

4. BEHAVIORAL EVIDENCE

The hypothesis in Section 3 is tested here using a setting in which every intermediate value is independently verifiable and every aspect of a model's behavior, that is, its weights, and its decoding, is reproducible by inspection.

The airline domain introduced in Section 2 is evaluated here under these conditions directly using Llama 3.1 8B Instruct [5], an open-weight, instruction-tuned large language model in 4-bit quantization, with greedy decoding throughout so that a given prompt deterministically produces the same output. A two-condition design isolates the distinction Section 3 defines. In the baseline condition, a model is given a problem and its governing rules and produces every intermediate value, elementary and boundary-dependent alike, before a final answer. In the boundary-isolation condition, the same problem and the same rules are given with every elementary value pre-filled, leaving only the boundary-dependent values, the fee a bracket assigns, for the model to determine. The rules and the correct final answer are held fixed across both conditions, so any difference between them is attributable to the removal of elementary-value burden specifically, not to an easier task overall. Figure 2 summarizes the two conditions. In the baseline condition, a model produces every value, including the dimension and weight totals as well as the bracket fee. In the boundary-isolation condition, the elementary values are pre-filled, leaving only the bracket fee, the derivation boundary, for the model to determine. The fee step is the one constant across both conditions.

ONLINE FIRST

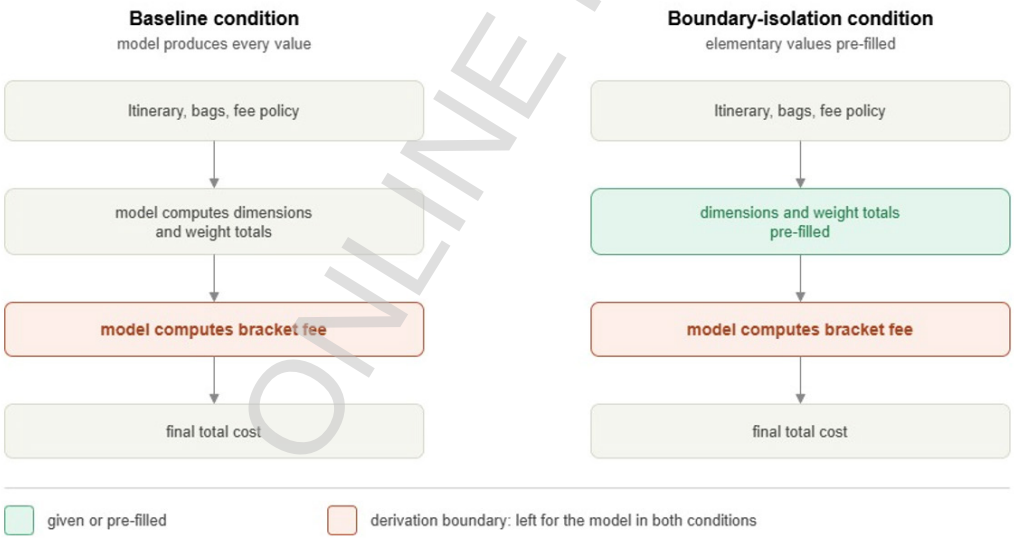


Figure 2. *The two experimental conditions compared: baseline and boundary-isolation.*

A correct final answer in either condition requires a long sequence of correct intermediate values. A single accuracy figure conflates two different kinds of error: a model can fail to match the final answer for reasons unrelated to a derivation boundary, or it can match the final answer by chance despite mishandling one. This concern is related to, but distinct from, findings that LLM accuracy on mathematical problems degrades broadly under surface-level numerical changes alone, with no localized account of where in a problem this degradation occurs [6]. The analysis below is conducted at the level of the individual

derivation step rather than the final answer alone, to test whether the error a single accuracy figure obscures is in fact concentrated at a specific kind of step rather than spread diffusely across a problem. At each derivation boundary, a model's output is recorded as one of three outcomes: the boundary-dependent value is correctly resolved; a directly available but rule-irrelevant value is substituted in its place; or a value with no traceable source in the input is produced, accompanied by language asserting or implying that it came from the rules. The last of these is fabrication and is distinguished from a related but distinct error described below.

Exact-match accuracy is low in both conditions. It is near zero in the baseline condition and only marginally higher in the boundary-isolation condition. On its own, this figure understates what the boundary-isolation condition reveals. Comparing each condition's predicted total against the ground truth, rather than checking only for an exact match, the boundary-isolation condition produces a prediction closer to the correct total in just over half of all cases, with none tied. Removing the burden of elementary computation does move a model's output closer to the correct answer in most cases, even though it rarely reaches it outright. This is consistent with elementary derivations contributing real error in the baseline condition while derivation boundaries continue to contribute error in both. No case in either condition produces a prediction wildly distant from the ground truth, the kind that would indicate a model abandoning the task or guessing without constraint. To isolate step-level accuracy from any error introduced elsewhere in a shared context, each step was additionally tested independently, given only the minimal information that the step requires, rather than as part of a full multi-step response. Across all sixty problems, this produced two thousand one hundred sixty individually scored steps, one thousand three hundred fifty-one elementary, and eight hundred nine boundary-dependent. Elementary steps were correct in ninety-seven percent of cases. Boundary-dependent steps were correct in fifty-eight percent of cases. The gap holds within the elementary category itself: steps requiring only a sum or a direct comparison against a stated threshold were correct in every case, while steps requiring a fee to be read off an already-given bracket dropped slightly, to ninety and ninety-two percent. The boundary category shows the same pattern more sharply, with steps that convert a raw measurement into a fee through a policy table ranging from fifty-two to seventy percent correct. Isolated from any surrounding context, the gap between elementary and boundary-dependent steps persists and tracks the same distinction Section 3 defines. The errors that remain are errors of a specific, recurring kind, examined next, rather than a general breakdown in the task's execution.

Two distinct error types account for nearly all the failures observed at derivation boundaries in this domain. The first is bracket merging. This occurs when a stated quantity sits near the line separating two adjacent brackets in a rule's schedule. The model treats the two brackets as a single wider range and assigns the fee belonging to the wrong one. For example, a bag weighing fifty-three pounds falls under the policy's own definition into a bracket assessing a fee of thirty dollars. But the model instead describes it as falling within a merged range spanning fifty to seventy pounds and assesses the fee belonging to the higher bracket. This is boundary misassignment in its clearest form: the model retrieves a real value from the rules but assigns it to the wrong side of a boundary it has not resolved precisely. Across sixty independently generated problems in this domain, this pattern recurs in sixteen.

ONLINE FIRST

The second error type is fabrication. This occurs when a model states that a value is not specified by the rules, when in fact it is, and substitutes an invented figure in its place. This pattern recurs with a lower frequency in the airline domain itself. But it reappears in a cleaner, more clearly diagnostic form in the tax domain introduced earlier. There, a tax-table lookup is twice resolved by stating that the table is not provided. A specific figure follows immediately after. The same justification recurs almost verbatim across two independently generated problems concerning two different taxpayers. Both errors share a feature that distinguishes them from an arbitrary mistake. Neither announces itself. The text surrounding a merged bracket or an invented table value reads no differently from the text surrounding a value correctly resolved. If anything, it is more elaborately justified, since a model writing around an unresolved value has more work to do to make the result look resolved.

The same two-part picture holds across domains: reliable elementary computation paired with unreliable boundary resolution. This appears in the airline domain under controlled, quantitative testing, and in the tax domain under qualitative inspection of individual cases. The industrial classification task discussed earlier showed the same pattern, though under neither condition, since that setting permitted no controlled comparison at all. Three domains sharing no vocabulary, no numeric structure, and in two cases no quantitative comparability with one another converge on the same answer to the question Section 1 opened with. Failures do not spread evenly across a model's reasoning. They concentrate on derivation boundaries.

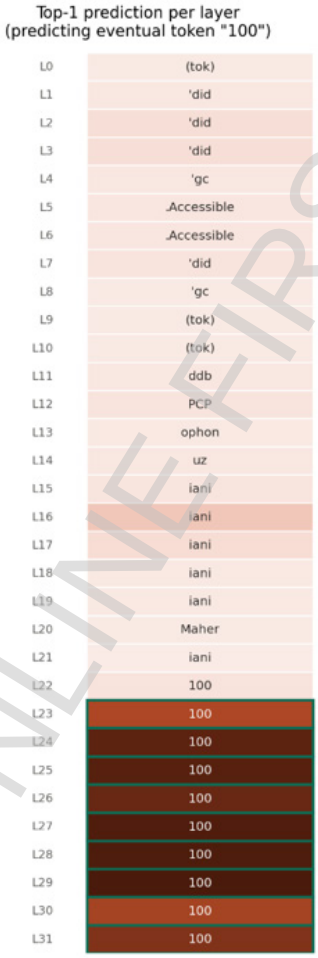
ONLINE FIRST

5. MECHANISTIC ANALYSIS

Section 4 establishes where failures occur: at derivation boundaries, identifiable in advance from a task's structure, and confirmed by a recurring, named pattern of error in the model's output. This leaves a separate question open. Knowing where a model fails in its output says nothing about what is happening inside the model at that point. What follows looks inside the model itself to trace this.

A model's output at any position is the result of a computation carried out across many layers, each transforming the representation it receives from the one before. At every layer, the current representation can be decoded. This means passing it through the model's own final normalization and output projection, the operation ordinarily applied only at the last layer, to see what the model would have predicted had generation stopped there [7]. Applied across all layers at a single position, this reveals not only what a model eventually outputs but also the layer at which its internal representation commits to that output, and whether anything else was still under consideration when it did. At a position where the model correctly restates a value given directly in its input, the flight ticket price of one hundred seventy-eight dollars stated outright in the passenger's itinerary, the trajectory across layers is simple. The correct value is not yet the leading candidate through the first twenty-two layers. At layer twenty-three it becomes dominant, at a probability of 0.636, and by layer twenty-seven it exceeds 0.99. No other value appears as a serious alternative at any layer after this point. A directly available value emerges early, stabilizes early, and remains stable for the rest of the network.

Before tracing where this uncertainty originates inside the network, it is worth noting that it is sometimes visible even in the model's own final output. At the point where Bag 2's overweight fee is generated, the model's output-layer distribution assigns the eventual token, $100, a probability of only 0.788, with the correct alternative, $30, still assigned 0.199, a genuine four-to-one split rather than a confident error. Later in the same response, recombining two bag totals, the model's final-layer distribution splits almost evenly, $130 at 0.515 against $300 at 0.485, a near coin flip at the very point where its answer is produced. This uncertainty is not visible in the text of the response itself, which states each figure with the same flat confidence throughout; it is only visible by inspecting the probability distribution that the model's own output layer assigns (Figure 3).



ONLINE FIRST

Figure 3. *Top 1 predicted token at each of the model's 32 layers for the position generating the bracket fee, where the eventual (incorrect) output is $100.*

At a position where the model produces a misapplied bracket value, the fee assigned to a bag weighing fifty-three pounds, a value the policy's own bracket structure assigns to thirty dollars but the model instead reports as one hundred dollars, the trajectory looks different in every respect. No coherent signal appears for the same first twenty-two layers. The value the model will eventually output appears for the first time at layer twenty-two, weakly, at a probability of 0.033, and only becomes dominant by layer twenty-five, at 0.952. Even past this point the correct value,

thirty dollars, remains a live alternative through the final layers, rising to 0.321 at layer thirty before the network settles again on the incorrect value at the output layer.

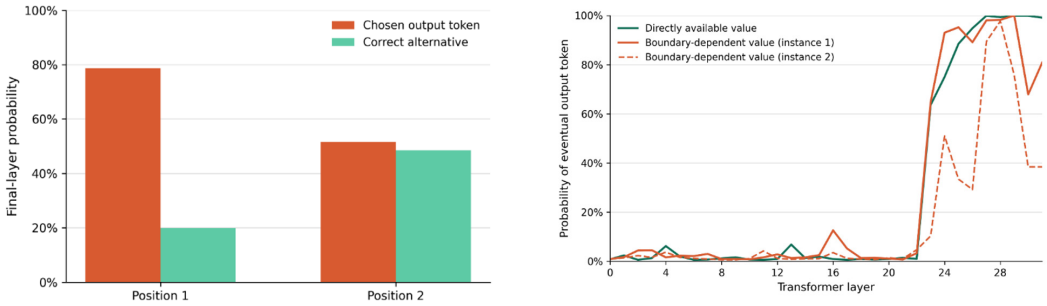


Figure 4. *Left: final-layer probability assigned to the chosen output token versus the correct alternative, at two positions in the same response. Right: probability assigned to the eventual output token across all 32 transformer layers, comparing a directly available value against two boundary-dependent value instances.*

A second instance, the same misassignment applied to a different bag in the same response, weighing fifty-one pounds, fluctuates more and competes more directly. The correct value, thirty dollars, becomes the dominant prediction twice before the network reaches its final output: briefly at layer twenty-five, at 0.51, and more strongly at layer twenty-eight, at 0.894, a higher confidence than the model ever assigns the value it eventually produces. Layers twenty-nine and thirty swing back toward the incorrect value, two hundred dollars, and at the final layer the three live candidates, two hundred, one hundred, and thirty dollars, sit at 0.384, 0.299, and 0.264, a near three-way tie resolved only by whichever candidate is narrowly ahead when a token must finally be chosen, the same late, fluctuating pattern shown as the second instance in Figure 4. A boundary-dependent value emerges late, fluctuates, and continues to compete with a correct alternative long after a directly available value would have settled.

Set against one another, these trajectories support a single claim. They are illustrative rather than a systematic sweep across all derivation boundaries in the dataset. Values requiring a rule-conditioned transformation, the values a derivation boundary depends on, are committed to later and less stably than directly available values. A directly available value is resolved early and is never seriously contested again. A boundary-dependent value resolves late, if it resolves at all in any stable sense. A correct alternative can remain genuinely live deep into the network. In one case, it remains stronger than the value the model finally outputs. This is the mechanistic counterpart to the behavioral asymmetry in Section 4. The values a derivation boundary requires are produced incorrectly more often. They are also settled on later in the computation, and less firmly, than values that crossed no such boundary.

The pattern above connects directly to the failure documented at the level of output text. A derivation boundary delays resolution. The representation needed to produce a boundary-dependent value is not yet settled by the layer on which most values have already stabilized. This delay leaves room for competing candidates. Several plausible values remain live at once, where a directly available value would have left none. Resolution still arrives, but late. The candidate it settles on is whichever was narrowly favored at that late

ON IN PRESS

stage, not necessarily the one a correct lookup would have produced. What reaches output is a late commitment, dressed in the same confident register as an early one. This chain is what surfaces as the fabrication and misassignment documented in Section 4. A derivation boundary leads to delayed resolution. Delayed resolution leaves room for competing candidates. A late, narrow commitment among them is what the model finally outputs. Section 4 left one question unanswered. Why a derivation boundary is specifically where this happens. That question is answered by what occurs inside the model, in the layers before an output is ever produced.

6. DISCUSSION

Section 4 answered where faithfulness failures occur. Section 5 examined what happens inside the network at those points. What follows considers what this means.

Several recent papers address related aspects of LLM unfaithfulness and error. Biran et al. (2024) show that large language models resolve two-hop factual queries by first resolving a bridge entity in early layers and then using it to resolve the second hop in later layers and introduce back-patching to recover cases where this pathway fails [12]. Shen et al. (2026) introduce a benchmark for evaluating chain-of-thought unfaithfulness at the level of individual instances, testing eleven detection methods across several models [13]. Zhu et al. (2026) find that reasoning errors concentrate at a small number of transition points marked by spikes in token-level entropy and introduce an inference-time method that intervenes at these points [14]. Yang et al. (2026) address the trade-off between faithfulness to retrieved knowledge and expressiveness of generated text, proposing a decoding method that balances the two [15]. Liu et al. (2026) study hallucinations in LLM-generated code, building a taxonomy of hallucination types specific to programming tasks [16].

Whether the kind of operation a reasoning step requires, restatement versus a rule-conditioned lookup, predicts where faithfulness failures concentrate has not been examined. Tested in isolation from any surrounding context, a rule-conditioned lookup was correct in fifty-eight percent of cases, against ninety-seven percent for steps requiring only restatement or direct computation.

A model can appear highly faithful and highly competent at the same time, and the account above explains why this is not a contradiction. Most of what a model is asked to do, in a typical reasoning chain, consists of steps where producing fluent text and producing a correct value are the same act: a restatement, a sum, and a comparison against something already given. A model's apparent reliability on these steps says nothing about its reliability at the comparatively rare steps where this coincidence breaks down. Chain-of-thought reasoning, in which a model produces step-by-step intermediate text before its final answer [8], is not unfaithful everywhere, varying substantially across tasks in how strongly a model's stated reasoning actually drives its answer [2], and treating it as a single property, faithful or not, obscures the more useful fact that unfaithfulness clusters at derivation boundaries specifically, points fixed by a task's structure rather than found by inspecting a generated trace.

If a derivation boundary can be identified from a task's structure before a model is ever run, verification does not need to treat every step of a reasoning chain as equally suspect. Effort can be directed specifically at the steps most likely to fail, such as a stated value

ONLINE FIRST

followed by a rule-conditioned lookup, rather than a stated value followed by a sum. The fabrication signature documented in Section 4, a stated value accompanied by language asserting it came from a rule that does not in fact specify it, is narrower still, and mechanically detectable without inspecting a model's internals at all. Verification aimed at derivation boundaries is a smaller problem than verification aimed at faithfulness in general.

A reward or preference signal computed over a complete output cannot distinguish a fabricated value from a genuinely derived one whenever the two happen to produce a similar final answer. It also gives no signal isolating the specific position in a trace where a derivation boundary was crossed. This limitation is one process supervision is designed to avoid, having already been shown to outperform outcome-level supervision in mathematical reasoning [9], and it is consistent with separate findings that LLMs do not reliably use their own intermediate reasoning steps when producing a final answer, which has motivated reward designs built around each step's causal contribution rather than its surface correctness alone [4]. A reward scoped to exactly these positions is a different proposal that is positive when a value traces to an explicit lookup or to an explicit statement that the value cannot be determined, and negative when a value is asserted as drawn from rules that do not support it. This is a reinforcement learning problem rather than a supervised one. The desired behavior at a genuine derivation boundary is not always a single string to imitate. It depends on whether the required value is recoverable from the rules provided. The faithful response is either the correct lookup or an honest statement that the value cannot be determined. Only a reward defined over the property of faithfulness itself, rather than over a fixed target, accommodates both. The pattern observed in Section 5 sharpens this. If a boundary-dependent value is genuinely undecided until late in the network, with a correct alternative still live at the point a token is chosen, a reward applied at exactly that position has something real to act on: a representation in which the right answer has not yet been ruled out.

The definition in Section 3 refers to a relationship between what is stated and what a rule requires, not to natural language specifically, and applies to any setting in which a system follows explicit rules over structured input, whatever form that input takes. Whether the same asymmetry recurs in such settings, including the same late, unstable commitment documented in Section 5, is a natural extension of what is shown here. It is not a claim the present evidence supports on its own. Nor do the cases examined here involve more than a single model call producing a single checkable answer. Many deployed systems chain model outputs into further actions, where one step's output becomes the next step's input without independent verification. This does not introduce a new failure so much as remove a safeguard against the one already documented. A derivation-boundary fabrication remains visible here because the full trace can be inspected against a known ground truth. In a chained setting, the same fabricated value is passed forward as though retrieved, and nothing downstream marks the difference.

7. CONCLUSION AND FUTURE WORK

Large language model faithfulness failures are not uniformly distributed across a reasoning process. They concentrate at derivation boundaries, the points at which a rule-conditioned value must be derived from a fact already known rather than restated or summed

ONLINE FIRST

from it. Layer-wise analysis suggests why such values remain unresolved until late in the network's computation, with a correct alternative often still live at the point a token must be chosen, which leaves them particularly vulnerable to fabrication and to misassignment across an adjacent boundary.

This account points toward a specific training intervention rather than a general call for improved faithfulness. A reward scoped to a derivation boundary's position, rather than computed over a complete output, can distinguish a fabricated value from a genuinely derived one and can reward an honest admission that a value cannot be determined, not only a correct lookup. Because it is defined over how a value was produced rather than over a fixed target string, it fits reinforcement learning rather than supervised fine-tuning, in the same spirit as existing work aligning LLM behaviour to human-specified objectives [11].

The pattern identified in Section 5 makes this more than a general appeal to better training: if a boundary-dependent value remains genuinely undecided until late in the network, with a correct alternative still competitive at the point a token is chosen, a reward applied at exactly that position has a real representational state to act on. This suggests reward shaping targeted at the layers and positions identified through the same layer-wise analysis used here and raises a question this paper does not test: whether training against such a reward moves the resolution of a derivation boundary earlier and makes it more stable, the way a directly available value is already resolved early and without contest. Testing this directly is the most direct extension of the present work.

FUNDING:
This research received no external funding.

INSTITUTIONAL REVIEW BOARD STATEMENT:
Not applicable.

INFORMED CONSENT STATEMENT:
Not applicable.

CONFLICTS OF INTEREST:
The author declares no conflict of interest.

ONLINE FIRST

REFERENCES

[1] M. Turpin, J. Michael, E. Perez, and S. Bowman, "Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting," in Advances in Neural Information Processing Systems, vol. 36, 2023.

[2] T. Lanham, A. Chen, A. Radhakrishnan, et al., "Measuring faithfulness in chain-of-thought reasoning," arXiv preprint arXiv:2307.13702, 2023.

[3] R. Zhou, W. Hua, L. Pan, S. Cheng, X. Wu, E. Yu, and W. Y. Wang, "RuleArena: A benchmark for rule-guided reasoning with large language models in real-world scenarios," in Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics, 2025.

[4] D. Paul, R. West, A. Bosselut, and B. Faltings, "Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning," in Findings of the Association for Computational Linguistics: EMNLP 2024, 2024.

[5] AI@Meta, "Llama 3.1 model card," Meta AI, 2024. Available: https://github.com/meta-llama/llama-models

[6] I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, and M. Farajtabar, "GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models," in International Conference on Learning Representations, 2025.

[7] nostalgebraist, "Interpreting GPT: The logit lens," LessWrong, 2020. Available: https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, F. Xia, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in Advances in Neural Information Processing Systems, vol. 35, 2022.

[9] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, "Let's verify step by step," in International Conference on Learning Representations, 2024.

[10] V. Sreekumar, "Time Series Classification for Paper Web Breaks in Print industry", Master's thesis, SRH Hochschule Heidelberg, 2026.

[11] L. Ouyang, J. Wu, X. Jiang, et al., "Training language models to follow instructions with human feedback," in Advances in Neural Information Processing Systems, vol. 35, 2022.

[12] E. Biran, D. Gottesman, S. Yang, M. Geva, and A. Globerson, "Hopping too late: Exploring the limitations of large language models on multi-hop queries," in Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2024.

[13] X. Shen, S. Wang, Z. Tan, L. Yao, X. Zhao, K. Xu, X. Wang, and T. Chen, "Faith-CoT-Bench: Benchmarking instance-level faithfulness of chain-of-thought reasoning," in International Conference on Learning Representations, 2026.

[14] W. Zhu, J. Zhang, L. Yu, K. Yue, and Z. Tang, "Dissecting failure dynamics in large language model reasoning," in Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics, 2026.

[15] C. Yang, Q. Si, L. Wang, and Z. Lin, "Breaking the trade-off between faithfulness and expressiveness for large language models," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 40, no. 40, 2026.

[16] F. Liu, Y. Liu, L. Shi, Z. Yang, L. Zhang, X. Lian, Z. Li, and Y. Ma, "Beyond functional correctness: Exploring hallucinations in LLM-generated code," IEEE Transactions on Software Engineering, 2026.

ONLINE-FIRST

ONLINE FIRST