Interactive Visualization of Chain-of-Thought Reasoning in Language Models

Oyku Hatipoglu
ETH Zürich

Gül Oymak
ETH Zürich

Victor-Eugen Zarzu
ETH Zürich

Oscar Garries Urbina
ETH Zürich
U. Politècnica de Catalunya

ABSTRACT

Large Language Models (LLMs) with reasoning capabilities are rapidly becoming the cornerstone of tomorrow's knowledge infrastructure, yet how they arrive at their answers remains opaque. Interpreting this models is particularly challenging when multi-step reasoning methods like Chain-of-Thoughts (CoTs) are involved. Existing explanation tools often fall short in presenting the reasoning paths of LLMs. To address this gap, we present an interactive visualization interface designed for CoT. Our tool allows users to inspect each reasoning step given a multiple choice question. We display the model choices and their corresponding uncertainties to help the user track the decision making progress. Additionally, users can interactively intervene and modify the reasoning paths and observe immediate effects on the outcome. Therefore, the they can experiment with alternative reasoning paths and get insights about the model behavior, robustness and failure cases. In this paper, we discuss design choices, the implementation of the interactive interface and potential applications.

1 INTRODUCTION

While Large Language Models (LLMs) are becoming increasingly more crucial in question answering systems, understanding how they arrive at their answers is important for both technical and ethical concerns. LLMs can perform successfully on many complex tasks; however, their reasoning process remains a mystery for many, raising questions about the robustness and trustiness of the models.

Chain-of-Thought (CoT) [14] prompting has been used as a technique to promote step-by-step reasoning to improve the ability of LLMs to perform complex reasoning. CoTs are able to provide an interpretable "chain" of the LLM reasoning by breaking down the decisions into smaller, more comprehensible steps. Yet, even with these intermediate steps, users still face the challenge of understanding how each decision influences the final outcome. To make the reasoning process more understandable, an interactive tool is needed to allow users to discover how CoT prompting works.

Existing CoT implementations present the chain with static text nodes and lack support for interacting with intermediate nodes. To address this, we introduce an interactive interface for visualizing and intervening with the CoT reasoning process. This way, we aim to enable interaction with each reasoning step and let users observe how this change reflects on the final outcome. Our tool also shows the confidence and entropy of the answers for each node to make the chain inspection easier while promoting transparency, reducing confusion, and increasing user trust. To achieve this, we designed an interactive visualization tool tailored specifically for CoT reasoning in multiple choice questions using LLMs.

We implemented our interface using React and TypeScript for the front-end and integrated the back-end with the Flask [4]. We used the Ministral 8B [9] model variation by Mistral AI as our LLM. This interactive interface aims to lower the barrier between complex model reasoning behavior of LLMs and humans to serve as a tool for debugging and refining reasoning in CoT-enabled language models. Additionally, this setup supports the explainability of model behavior and opens up new possibilities for education and user-in-the-loop alignment.

2 RELATED WORK

2.1 Explainability in LLMs

Explainable Artificial Intelligence (XAI) focuses on designing methods that make the behavior and decisions of Artificial Intelligence (AI) models that are seen as "black boxes" more transparent and interpretable [11]. Traditional XAI methods like saliency maps [12] or feature attribution methods [6] usually work best with vision models and are not suitable for LLMs due to the scale of the model, autoregressive generation and lack of intermediate representations like visuals. Alongside explaining black-box decisions, XAI methods can also make user understanding stronger and improve human-AI collaboration when using LLMs.

According to Bilal et al. [1], explainability in LLMs is divided into three categories: (1) post-hoc explanations, (2) intrinsic interpretability and (3) human-centered explanations. Post-hoc techniques like SHAP and LIME aim to explain the model's decisions by analyzing the correlation of input question and output answer. Intrinsic interpretability focuses on prompting techniques that can show their reasoning process such as Chain-of-Thought (CoT) prompting. Finally, human-centered methods aim to generate explanations regarding the user needs, using interaction or feedback mechanisms to ensure transparency and trust. Additionally, LLMs themselves can improve on XAI methods through natural language generation, automatic explanation and human-like evaluation [15]. All these strategies aim to enlighten the "black box" of LLMs while also constructing explanations that support real-world usage, inspection and debugging of LLMs.

2.2 Chain-of-Thought Reasoning

Chain-of-Thought (CoT) reasoning is a prompting technique that is used to guide LLMs to solve problems by generating intermediate reasoning steps before giving a final answer. Introduced by Wei et al. [14], CoT improves performance on complex tasks including arithmetic, logical reasoning and multiple choice questions. It breaks down the problem into a sequence of logically connected steps and forms a chain of thought, just like how a human thinks to get to the solution.

While it improves the performance of LLM answers, CoT reasoning also significantly enhances the interpretability of LLMs. Recent work by Wang et al. [13] introduced Chain-of-Probe (CoP) to systematically evaluate reasoning confidence for each step. Their findings also revealed that CoT's performance vary with task complexity and the correctness of each intermediate step must be verified seperately, it is not enough to investigate the confidence of final prediction.

While the CoP method underlines the importance of verifying intermediate steps, our approach extends beyond verification by enabling direct user interaction and allowing real-time modifications of reasoning paths to actively enhance understanding and reliability.

Existing visualization tools often limit user interactions to passive viewing [17]. Unlike other previous tools, our interface motivates the users to explore and grasp the reasoning processes of LLMs by enabling active exploration and modifications to bridge the gap between passive observation and active experimentation in explainable AI.

2.3 Applications and Interfaces for CoT

As CoT prompting improves model performance and also transparency, its applications also expand across difference domains that demand context-dependent reasoning. Beyond the standard benchmarks that use arithmetic, logic and multiple choice questions, CoT is also being integrated into real-world scenarios that benefit from more interpretable model behavior.

In the medical domain, CoT prompting has shown promising results in aligning LLM reasoning with human-like accountability and decision making. For example, Miao et al. [7] explored the use of CoTs in nephrology, showing how it strengthens the specificity, ethical alignment with humans and comprehensibility of LLM responses. The authors argue that CoT helps to transform the black-box AI decisions to transparent reasoning expected by medical professionals. Since this is a high-risk domain that concerns humans health, it is important to see CoTs as critical components for making LLMs more trustworthy in healthcare and further sensitive scenarios.

Another application of CoTs include being adapted to improve explanation quality in interactive dialogue systems. Xu et al. [16] introduce Chain-of-Thought Explanation (CoTE) for Dialogue State Tracking (DST), where models generate intermediate reasoning steps to justify slot value predictions. By integrating CoT into generative DST pipelines, they achieve not only improved accuracy but also more readable and interpretable outputs.

While these different applications show how CoT can be used for transparency, they usually present explanations in static form. Our work builds an interactive interface for CoT reasoning by enabling the users to edit, prune and experiment with different reasoning paths in real time. We aim to transform the CoT from a passive explanation to a tool for active exploration.

3 User Interface

In the following, we describe the layout, visual components, and interaction flow of our tool. An overview is illustrated in Figure 1.

3.1 Node Operations

Node operations provide users with direct and flexible control over the Chain-of-Thought tree. Users can dynamically edit, introduce random interventions, or selectively hide or prune branches by interacting with individual nodes. An example node with all its buttons can be seen in Figure 2. Node operations enable exploration and allow localized modification of reasoning paths and are meant to form the core of human-model interaction within the interface. A visual explanation can be seen in the Appendix B.

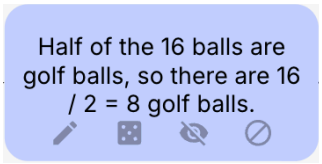


Figure 2: Node example.

3.1.1 Edit

Clicking the edit button on a node of choice allows users to change its text directly. A popup window appears, users type in their desired edits and save them. Once the edits are saved, a new branch with the edited content appears alongside the original node. Keeping the original node visible helps users compare the modifications clearly and observe how their edits influence the reasoning flow. This gives users the ability to shape the reasoning process and immediately observe the impact of their changes.

3.1.2 Random Intervention

When users click the random intervention button on a specific node, a new branch appears with a randomly modified version of that node. This creates a new reasoning path without manual edits. The original node also remains visible to make it easier for users to quickly explore alternative reasoning paths.

3.1.3 Hide/Show

The hide button on the node allows users to temporarily hide branches starting from the interacted node, reducing visual clutter and focus on specific reasoning paths. Hidden nodes can be revealed with the show button, which replaces the hide button in the same node when that node is hidden by the user.

3.1.4 Prune

When the prune button is pressed on the node, the corresponding branch starting from that node will be removed from the interface. This will allow users to eliminate useless or logically flawed reasoning paths.

3.2 Header Operations

The header panel gives users global control over the generation and exploration process. It consists of a return back button that can be used for re-entering another question and the following three CoT operations:

3.2.1 Stop Generation

The Stop Generation button stops the CoT generation process at any point. This functionality is useful for stopping the generation of long or unwanted branches.

3.2.2 Random Intervention

The Random Intervention button on the header functions similarly to the Random Intervention button on the node. The difference is, the intervened node is also selected randomly. When pressed, it automatically applies a randomized modification to a randomly selected node, which encourages rapid exploration of alternative reasoning paths. It also increases the usability of our interface as it removes the burden of choosing which node to modify from the user.

3.2.3 Regenerate CoT

The Regenerate CoT button resets the current CoT tree and prompts the model to generate a new CoT from scratch. This functionality is particularly useful when the workspace is overly crowded and the user wants to restart the CoT generation.

3.3 Design Decisions

In choosing a tree layout over a simple linear or list-based presentation, our goal was to mirror the branching nature of Chain-of-Thought itself. A list forces users to follow a single path while hiding alternative hypotheses. In contrast, a tree makes parallel reasoning paths explicit and allows users to visually compare multiple chains in one glance and spot where splits occur. This spatial organization also reduces the cognitive load when tracing back from a leaf node to its common ancestor, since parent-child relationships are drawn directly rather than implied by list indices.

To render the tree layout, arrows, and support pan/zoom interactions, we use the D3.js library [10]. React manages the state of

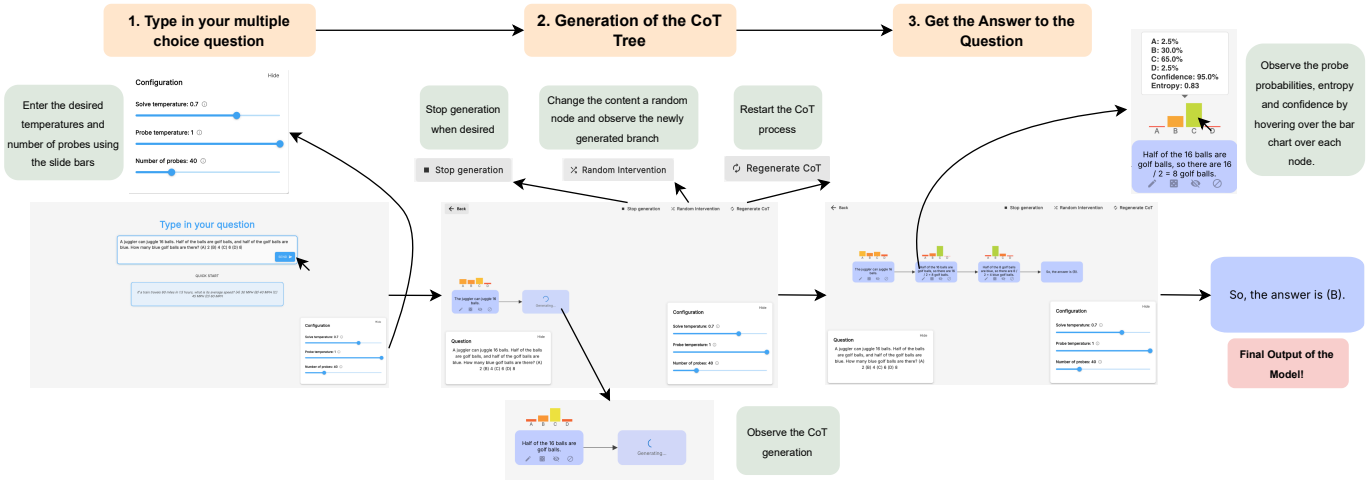


Figure 1: Main Interface Flow

the interface and the component structure, while D3 handles the SVG-based rendering of the reasoning branches.

To convey the uncertainty of the model at each step, we embed a small bar chart of entropy directly within each node rather than relegate statistics to a separate panel. This inline encoding couples confidence information with the text it qualifies, minimizes eye movement, and helps users detect low-confidence junctures on the fly.

When users edit a node, we keep the original version visible along with the new branch rather than overwrite it. This parallel display supports direct comparison of "before" and "after" reasoning steps, which is critical for debugging: one can immediately see the effect of a reworded premise on downstream choices and confidence. Pruning or hiding remains complementary; edits generate new hypotheses, while hide / prune lets you silence or remove those you have already evaluated as unhelpful.

Header-level controls (global random interventions, stop generation, regenerate CoT) coexist with node-level tools to accommodate both broad and local exploration strategies. A user who wants to rapidly sample any alternative path can hit the header's random intervention button, whereas someone debugging a specific step can locally inject or prune just that branch.

4 Implementation

We start with the setup and model that was used. As the base LLM, we used the 8B variant of the Ministral model for all the computation through API calls to La Plateforme [8]. We present below the endpoints of features of the server built using REST principles [3].

4.1 Chain generation

In order to offer to the user and iterative way of visualizing the progress, every step is generated by prompting the model and instructing it to generate the next step given the prompt and the already generated steps. So, if at a given time we have $\text{step}_1, \text{step}_2, \ldots, \text{step}_n$, the prompt to generate step_{n+1} will contain the initial question and the first n steps. To get the answer in the desired format, we use a few-shot prompting [2] . We further force the model to give the final answer in a form of

$$\text{So, the answer is } (x). \tag{1}$$

after all the reasoning was done in the previous steps and where x is one of the multiple choice possible answers. This is done to clearly visualize the answer to which the chain converged.

Furthermore, along with the new step of the chain, the generated text also contains the confidence of the model in it's response. Lastly, since the current logic would generate an infinite chain, it need to be stopped after the answer is returned. For this, the prompt is designed in such a way that, if in the previous steps there is already an explicit answer like 1, the model has to return the text "Done." acting like a stop signal. This way, the same logic can be used to generate both the initial chain and the further variations produced by editing or random interventions. The temperature used for generated the chain can be provided in the request.

4.2 Probing

Probing refers to computing the probabilities of every possible answer, providing the question and a list of already generated steps. However, interacting with the model is done through API calls, so the probabilities of the tokens are not exposed. To overcome this, we computed the probabilities of each answer by prompting the model x_{probe} times and instructing it to generate just choice corresponding to the answer with few-shot prompting. The prompt used for choice generation can be found in Appendix A. This is followed by taking the probability of every choice and further computing the entropy of the distribution. The x_{probe} value and the temperature for returning the answer choices which is different from the solve temperature can both be passed in the request.

La Plateforme allows up to 10 responses from the model for a given prompt with a timeout of 1 second between requests. In order to allow a variable number x_{probe}, we use multiple API keys and run requests in parallel using threads, followed by aggregation of the results.

4.3 Random Intervention

The randomized modification is also performed using the same LLM. We again use few-shot prompting and interrogate the model to alter a given reasoning step, given the initial question and the previous chain steps. The question and previous steps are added for question and reasoning related context such that the result lies in the same area/domain. The temperature used for this intervention is the same as the one using for generating the steps of the CoT.

5 Use Case

Our interactive Chain-of-Thought visualization interface is designed to serve a diverse set of users and scenarios. Below, we outline three

primary use cases and illustrate how our tool adapts to each context.

5.1 Expert Analysis and Model Debugging

Machine learning practitioners and researchers often need deep insight into how a language model arrives at its conclusions. Our tool helps experts to:

Inspect detailed reasoning paths. Experts can step through each CoT node, observe its generated text alongside inline entropy bars, and identify which intermediate inferences carry high uncertainty or potential error. This fine-grained visibility supports root-cause analysis when the model fails on edge cases or produces contradictory chains.

Experiment with interventions. By editing a single step or injecting a random alternative, researchers can quickly probe the model's sensitivity to different premises. For example, changing a numeric assumption in step 3 and watching how downstream confidence and final choice shift exposes fragile links in the reasoning process.

Compare model variants. When evaluating multiple LLM checkpoints or hyperparameter settings, experts can load two chains side by side and contrast confidence profiles and branch structures. This side-by-side comparison accelerates selection of the most robust model for deployment.

5.2 Educational Exploration and Learning

For students, instructors, and non-expert users curious about AI reasoning, our interface acts as an interactive sandbox:

Transparent step-by-step walkthroughs. Learners see the model decompose a problem into discrete steps, mirroring pedagogical approaches in math and logic courses. Watching the chain grow helps determine how LLMs "think", reinforcing connections between prompts, intermediate inferences, and final responses.

Hands-on experimentation. Students can edit a reasoning node to test "what if" scenarios, e.g., altering a premise in a word problem, and immediately observe the effect in the following steps. This trial-and-error model fosters active learning, much like manipulating variables in an educational simulation.

Confidence cues. Visualizing uncertainty metrics helps learners understand where and why a model may struggle.

5.3 Human–AI Collaborative Reasoning

In many real-world settings, end users need AI outputs that align with domain knowledge or organizational requirements. Our tool supports human-in-the-loop refinement:

Dissect unsatisfactory outputs. When the final answer is incorrect or incomplete, domain experts can trace back through the chain, identify flawed logic, and manually prune or correct those nodes. This targeted intervention ensures that the AI's reasoning matches real-world constraints.

Customize reasoning flow. Users can hide irrelevant branches or rearrange key steps to tailor the CoT structure to their workflows, whether in legal analysis, financial forecasting, or policy development. This flexibility enables personalized reasoning paths that better reflect stakeholder priorities.

Iterative refinement. Users can regenerate the CoT at any stage or apply broad random interventions from the header panel, enabling rapid cycles of hypothesis and correction until the answer meets their expectations.

6 Limitations and Future Work

By using API calls due for interacting with the model due to computational resource constraints restricts our access to the log probabilities of the tokens. This forced using alternative methods for computing the probabilities of the answers after every step and the confidence of the model in every step, which can lead to inaccuracies in some cases (e.g. the answer probabilities computed using token probabilities are much different than the ones computed by prompting and averaging). An area of future work would be to switch to a locally running model (e.g. a Gemma3 [5] variant) and directly use the log probabilities.

Another limitation lies in the area of question types. Right now, the supported questions are the ones that have 4 available choices: A, B, C and D. One possible extension would be to either let the user set the available answer choices or infer them directly from the question by using the LLM. The latter is prone to instabilities due to model's error. Another extension would be to support open-ended questions (without the possible answer choices) and return the probabilities for the top k answers of the model. This would allow the interface to generalize beyond multiple-choice questions and be applied to a broader range of open-ended or free-form queries.

Furthermore, our interface currently focuses on text-based reasoning steps and does not incorporate other modalities, such as images or structured data, which may be present in real-world tasks. Integrating support for multimodal inputs and reasoning chains could greatly extend the applicability of the tool. Another possible direction is to include collaborative features that allow multiple users to annotate, edit, or discuss reasoning chains, which could be valuable in educational or research settings.

Lastly, while we display model confidence through entropy and step-level probabilities, we currently do not visualize or track the evolution of reasoning quality over time. Adding functionality to assess coherence, detect logical fallacies, or visualize the semantic progression of reasoning could further enhance the tool's value for both debugging and pedagogy.

7 Conclusion

In this paper, we presented an interactive visualization tool designed to enhance the interpretability and usability of reasoning processes in Large Language Models, with a particular focus on multi-step Chain-of-Thought reasoning. This kind of reasoning has become increasingly important as LLMs are applied to more complex tasks where understanding the intermediate steps is crucial. Although current prompting strategies, such as step-by-step guidance, can make these models more interpretable, users still often struggle to trace how individual decisions contribute to the final output. Our tool addresses this challenge by offering an environment in which each step in the reasoning process can be explored, edited, and analyzed interactively.

The interface goes beyond passive observation by allowing users to modify and experiment with individual reasoning steps, which helps in understanding how changes influence the overall logic and outcome. We also include visual cues for confidence and uncertainty, making it easier to spot where the model may be uncertain or prone to error. This helps users identify patterns of reliable or faulty reasoning and supports more informed decisions when working with model outputs.

By supporting such interaction, our tool serves a wide range of users: from researchers and developers who need to debug and improve model behavior, educators and learners seeking to understand AI reasoning better, and end users aiming to align model outputs with specific goals. This kind of analysis is not just about making AI decisions more transparent; it is also about empowering users to play an active role in shaping and understanding the reasoning process.

Ultimately, our tool bridges the gap between opaque model behavior and human interpretability by making Chain-of-Thought reasoning processes visible, modifiable, and explorable. This kind of interaction encourages users not only to understand model decisions, but also to guide and improve them, marking a step forward in building more transparent and collaborative AI systems.

REFERENCES

[1] A. Bilal, D. Ebert, and B. Lin. Llms for explainable ai: A comprehensive survey, 2025.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. CoRR, abs/2005.14165, 2020.

[3] R. T. Fielding and R. N. Taylor. Architectural styles and the design of network-based software architectures. PhD thesis, 2000. AAI9980887.

[4] Flask Team. Flask Framework User Guide.

[5] A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, L. Rouillard, T. Mesnard, G. Cideron, J. Grill, S. Ramos, E. Yvinec, M. Casbon, E. Pot, I. Penchev, G. Liu, F. Visin, K. Kenealy, L. Beyer, X. Zhai, A. Tsitsulin, R. Busa-Fekete, A. Feng, N. Sachdeva, B. Coleman, Y. Gao, B. Mustafa, I. Barr, E. Parisotto, D. Tian, M. Eyal, C. Cherry, J. Peter, D. Sinopalnikov, S. Bhupatiraju, R. Agarwal, M. Kazemi, D. Malkin, R. Kumar, D. Vilar, I. Brusilovsky, J. Luo, A. Steiner, A. Friesen, A. Sharma, A. Sharma, A. M. Gilady, A. Goedeckemeyer, A. Saade, A. Kolesnikov, A. Bendebury, A. Abdagic, A. Vadi, A. György, A. S. Pinto, A. Das, A. Bapna, A. Miech, A. Yang, A. Paterson, A. Shenoy, A. Chakrabarti, B. Piot, B. Wu, B. Shahriari, B. Petrini, C. Chen, C. L. Lan, C. A. Choquette-Choo, C. Carey, C. Brick, D. Deutsch, D. Eisenbud, D. Cattle, D. Cheng, D. Paparas, D. S. Sreepathihalli, D. Reid, D. Tran, D. Zelle, E. Noland, E. Huizenga, E. Kharitonov, F. Liu, G. Amirkhanyan, G. Cameron, H. Hashemi, H. Klimczak-Plucinska, H. Singh, H. Mehta, H. T. Lehri, H. Hazimeh, I. Ballantyne, I. Szpektor, and I. Nardini. Gemma 3 technical report. CoRR, abs/2503.19786, 2025. doi: 10.48550/ARXIV.2503.19786

[6] S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions, 2017.

[7] J. Miao, C. Thongprayoon, S. Suppadungsuk, P. Krisanapan, Y. Radhakrishnan, and W. Cheungpasitporn. Chain of thought utilization in large language models and application in nephrology. Medicina, 60(1), 2024. doi: 10.3390/medicina60010148

[8] Mistral AI. Introducing la plateforme. `https://mistral.ai/news/la-plateforme`, 2023.

[9] Mistral AI. Un ministral, des ministraux. `https://mistral.ai/news/ministraux`, 2024.

[10] Observable. D3. `https://d3js.org`, 2011.

[11] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

[12] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

[13] J. Wang, Y. Ge, Z. Shan, and Z. Ji. Chain-of-probe: Investigating reasoning via confidence in chain-of-thought. arXiv preprint arXiv:2402.12716, 2024.

[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35:24824–24837, 2022.

[15] X. Wu, H. Zhao, Y. Zhu, Y. Shi, F. Yang, L. Hu, T. Liu, X. Zhai, W. Yao, J. Li, M. Du, and N. Liu. Usable xai: 10 strategies towards exploiting explainability in the llm era, 2025.

[16] L. Xu, N. Peng, D. Zhou, S.-K. Ng, and J. Fu. Chain of thought explanation for dialogue state tracking, 2024.

[17] Z. Zhou, Z. Zhu, X. Li, M. Galkin, X. Feng, S. Koyejo, J. Tang, and B. Han. Landscape of thoughts: Visualizing the reasoning process of large language models, 2025.

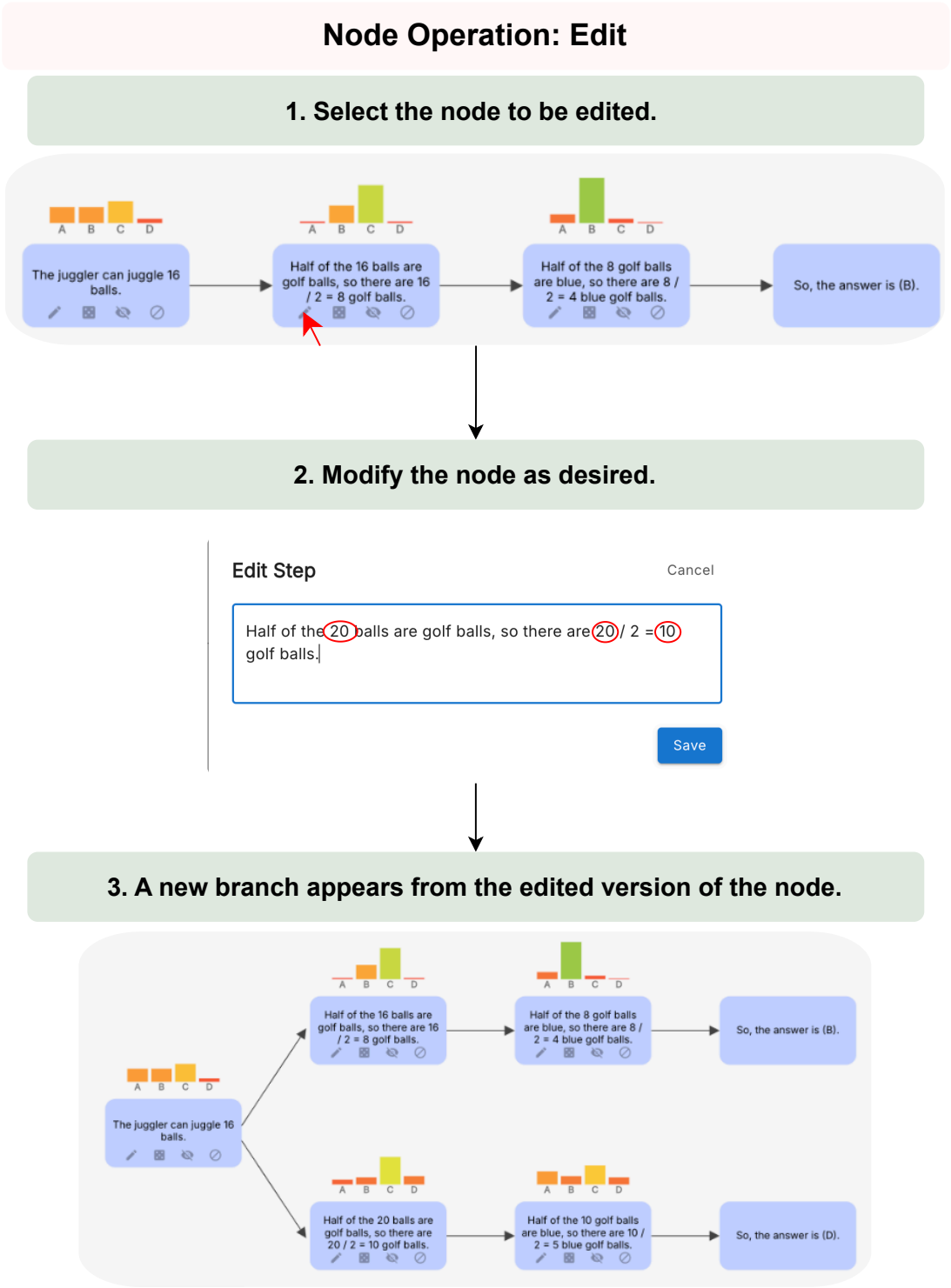
A PROBING THROUGH PROMPTING

For getting the probabilities of the answers given a list of generated steps, we used few-shot prompting by asking the model to generate the return one of the possible answer choices using the prompt showcased below in Listing 1.

```
{
"role": "system",
"content": "Generate the answer to the question, given the question and the previous reasoning steps. "
"Return just the choice letter of the answer."
},
{
"role": "user",
"content": """
Given the prompt and the previous reasoning steps, return the correct answer choice.

Here are some examples of questions with the previous steps and the generated next step:
Question: A cat has 4 legs. Three quarters of them are brown. How many brown legs does the cat have? (A) 1 (B) 2 (
    C) 3 (D) 4.
Previous generated reasoning steps: ['The cat has 4 legs.', 'Three quarters of the legs are brown.']
So, the answer is (
<Answer>
D

Question: There are 20 cars on the street. Half of them are white and half are blue. 60 percent of the white ones
    are Volvo. How many white Volvo cars are on the street? (A) 10 (B) 4 (C) 20 (D) 6.
Previous generated reasoning steps: ['The street has 20 cars on it.', 'Half of the cars are white.', 'So there are
     20 / 2 = 10 white cars.']
So, the answer is (
<Answer>
D

Question: There are 5 kids. 40 percent of them are girls. Half of the girls is blonde. How many blonde girls are?
    (A) 3 (B) 1 (C) 5 (D) 4.
Previous generated reasoning steps: []
So, the answer is (
<Answer>
B

Question: There are 12 cubes. Three quarters of them are yellow. How many yellos cubes are there? (A) 6 (B) 4 (C)
    3 (D) 9.
Previous generated reasoning steps: ['12 cubes exist.', 'Three quarters of them are yellow.', 'So there are 12 /
    (3 / 4) = 9 yellow cubes.']
So, the answer is (
<Answer>
D

Question: There are 20 cars on the street. Half of them are white and half are blue. 40 percent of the white ones
    are Volvo. How many white Volvo cars are on the street? (A) 4 (B) 10 (C) 20 (D) 6.
Previous generated reasoning step(s): ['The street has 20 cars on it.', 'Half of the cars are white.', 'So there
    are 20 / 2 = 10 white cars.', '60 percent of the white cars are Volvo.', 'So there are 10 * 0.6 = 6 white
    Volvo cars.', 'So, the answer is (D).']
So, the answer is (
<Answer>
A
"""
Question: {prompt}
Previous generated reasoning step(s): {steps}
Return just on letter, the choice of the answer to the question ({possible_choices}). So, the answer is ("""
}
```

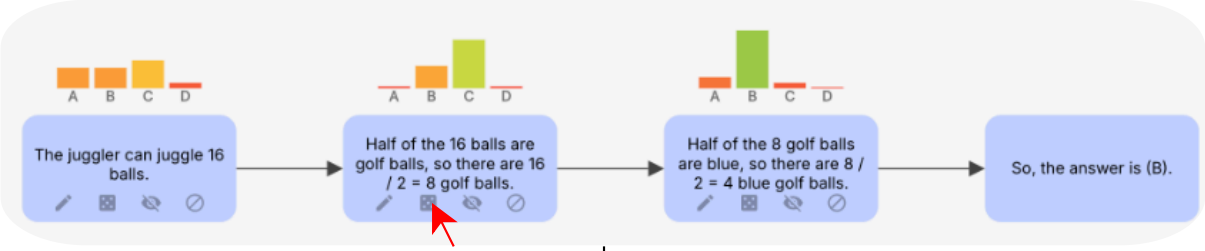
Listing 1: Few-shot prompting template used for probing.

B Visual Descriptions of Interface Features

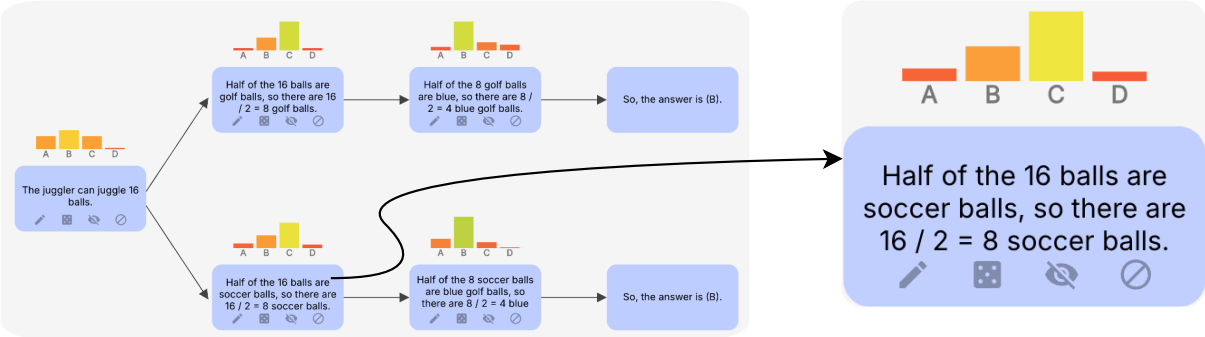


Node Operation: Random Intervention

1. Select the node to be randomly intervened.

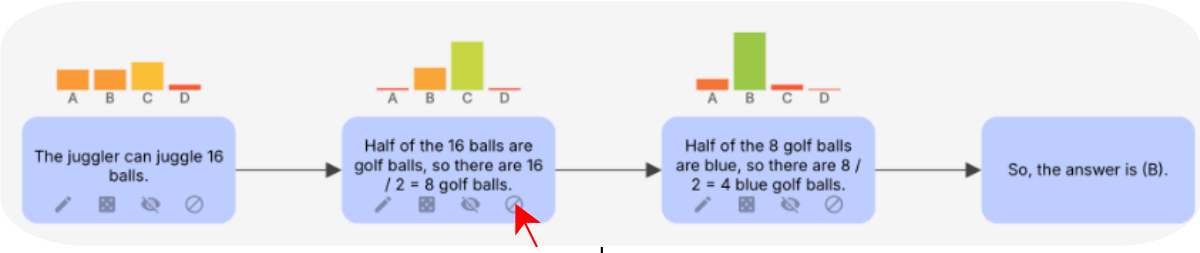


2. A new branch appears from the randomly intervened version of the node.

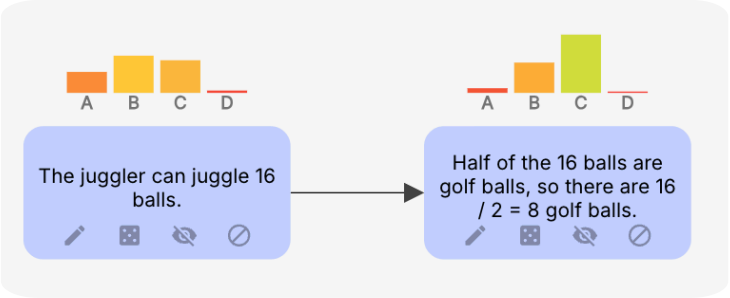


Node Operation: Prune

1. Select the branch to be pruned

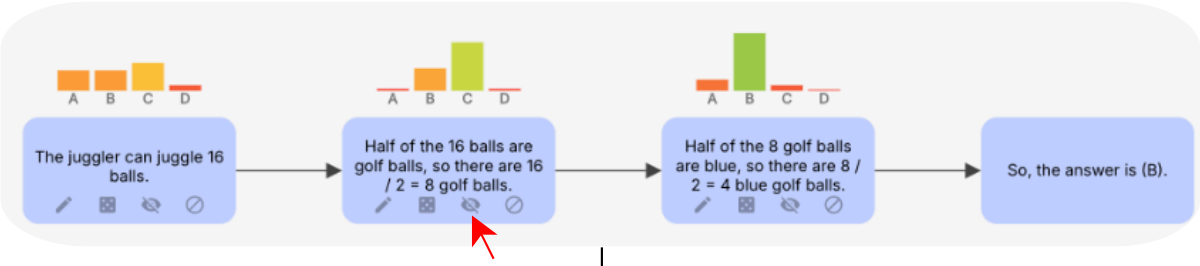


2. All subsequent branches are permanently removed.

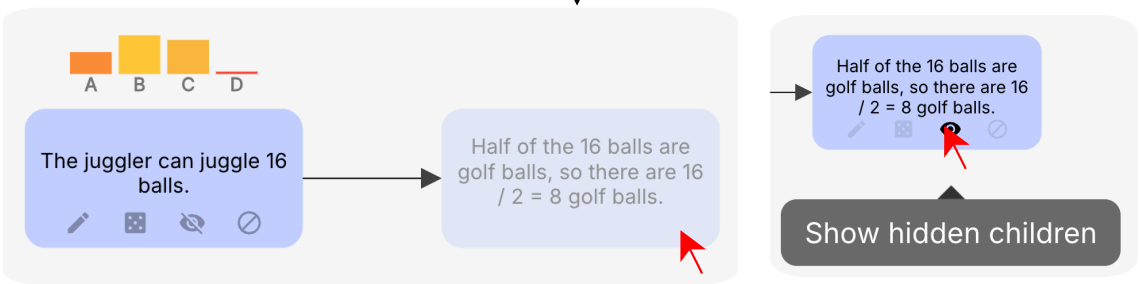


Node Operation: Hide

1. Select the branch(es) to be hidden.



2. All subsequent branches from the selected node are hidden from sight.



3. Hidden branches can be revealed by clicking "Show Hidden Children".