Tool Call Dependency Graphs Enable Deep LLM Reasoning Evaluation and Better Explanations

Nick Ferguson , **Alan Bundy** and **Kwabena Nuamah**
School of Informatics, University of Edinburgh
{nick.ferguson, a.bundy, k.nuamah}@ed.ac.uk,

Abstract

Evaluation of tool-augmented Large Language Models (LLMs) has not advanced far beyond final answer accuracy, and neglects in-depth evaluation of reasoning ability despite it being a central claim of recent models. We aim to address this gap by developing a dependency graph-based evaluation to give insight into meta-level reasoning ability. We create a two-stage algorithm to construct dependency graphs from a series of LLM-generated tool calls: in the first stage creating nodes for tool calls, then in the second creating edges where tool call results are identified as being used as arguments in subsequent calls. The resulting graph facilitates the development of graph-theoretic metrics which allows deeper evaluation of the reasoning ability of LLMs over the task in question. We also use the dependency graph to assist with the generation of natural language explanations of the question-answering process. We create a baseline explanation generated from the list of tool calls, and a TCDG-augmented explanation which incorporates information about information provenance. A human evaluation study shows that our TCDG-augmented explanations are preferred across three key explainability criteria, although the extra detail contained impacts ease of understanding.

1 Introduction

Large Language Models (LLMs), [Brown *et al.*, 2020; Radford *et al.*, 2019; Devlin *et al.*, 2019] have evolved from general purpose text-generation models to hybrid systems with claims of advanced reasoning ability. While earlier LLMs required greater efforts to elicit performance at more complex tasks [Wei *et al.*, 2022; Wang *et al.*, 2023b] *inter alia*, reasoning ability is now a central claim of recent LLMs [Llama 3 Team, 2024; Abdin *et al.*, 2024; Gemini Team, 2025]. Such models are now frequently augmented with capability to use external tools [Mialon *et al.*, 2023; Gao *et al.*, 2023; Parisi *et al.*, 2022], which enables well-known weaknesses of LLMs such as arithmetic reasoning and data fabrication to be alleviated through interaction with reliable & interpretable symbolic programs. Similarly, code generation models have been shown to provide a versatile problem-solving approach to many reasoning benchmarks [Li *et al.*, 2022].

Despite these advances, a major outstanding problem is the evaluation of the reasoning ability itself. Claims of the advanced reasoning ability of recent models is frequently inferred using basic metrics, such as final answer accuracy on popular multi-hop reasoning benchmarks like GSM8k [Cobbe *et al.*, 2021] or StrategyQA [Geva *et al.*, 2021]. While the term *reasoning* may take many forms, we focus on the problem of evaluating *meta-level reasoning* [Bundy, 1983], which can be understood as the problem of reasoning about the intermediate steps required to find a solution to a problem, and utilising appropriate object-level processes to complete such steps. We aim to address this gap by using the tool-calling paradigm as a lens through which we view the problem of evaluating meta-level reasoning. Tool calls serve as an intermediate representation of a meta-level reasoning process: interpretable, discretised steps which follow from a multi-step natural language plan.

Evaluating the ability of LLMs to utilise tools to complete a task may take different forms depending on the difficulty or requirements of the task. For example, when evaluating the correctness of a single tool call, simple accuracy metrics may be appropriate. While this may indicate whether a model can select the correct tool in constrained task settings, a different approach is needed to understand LLM tool use (and by extension, reasoning) ability for more complex tasks; multiple conversation turns; and use of a variety of different tools.

To address the problem of evaluating and explaining the reasoning ability of LLMs with tool use, in this paper we propose a reference-free method (i.e., not reliant on example solutions) for generating dependency graphs from a series of LLM-generated tool calls, which we use to define three graph-theoretic metrics aimed at providing tangible measures of meta-level reasoning quality. Additionally, we use information contained in the structure of the graph to improve natural language explanations of the quality of meta-level reasoning undertaken in the question-answering process, enabling better faulty reasoning detection. Our hypotheses are:

1. Creating a directed acyclic graph from a series of LLM-generated tool calls provides a versatile platform that enables the use of graph-theoretic metrics to obtain deeper empirical analysis of the reasoning process compared to basic final answer accuracy.

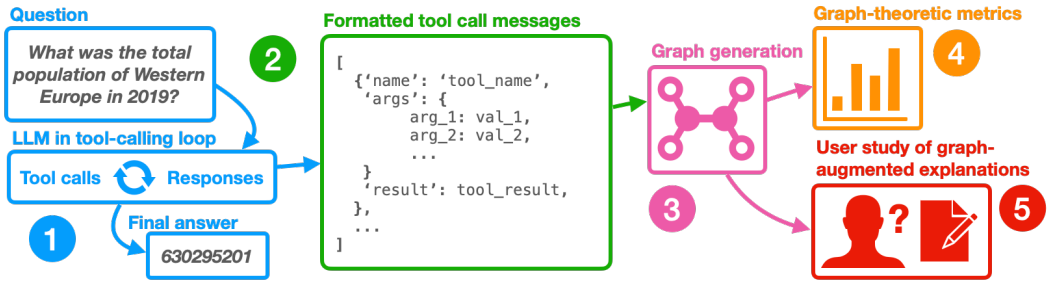

Figure 1: Overview of our graph generation and evaluation process. 1 We evaluate a range of LLMs on a multi-hop tabular QA dataset. (§3.1) 2 The list of tool calls and results is extracted into a single formatted list. (§3.1) 3 Dependency graphs which model tool use are created using our algorithm.(§3.2) 4 Graph-theoretic metrics are designed to quantify reasoning performance (§3.3). 5 Using the graph, natural language explanations are created and evaluated using human participants. (§4)

2. Information contained in such graphs improves the explanations of the meta-level reasoning process.

In addressing these, we make the following contributions:

- A reference-free algorithm for creating dependency graphs from a series of LLM-generated tool calls, with each node representing a tool call, and each edge representing the use of one tool's result as the argument to a subsequent call.
- Three proposed graph-theoretic metrics which demonstrate the depth of evaluation and insight into reasoning ability which is enabled by the graph structure.
- A human evaluation study showing that human participants find natural language explanations incorporating information from the graph structure useful for judging the quality of the answer produced by the LLM.

2 Background & Related Work

Evaluation of tool augmented language models has attracted much attention, with numerous datasets, benchmarks, and fine-tuned tool-use models released over recent years. However, despite innovations in the training of LLMs to utilise tools, evaluation of such approaches tends to be limited to accuracy, pass rate, or substring matching metrics. Substring matching over k generated words has been employed [Schick *et al.*, 2023], even if over real, human-generated references [Patil *et al.*, 2025; Patil *et al.*, 2024]. Pass and win rates using an LLM-based evaluator can quantify preference for candidate approaches [Qin *et al.*, 2024]. Domain experts to hand-create solutions over which a similarity score is computed between the LLM-generated tool calls [Lu *et al.*, 2025], and plain accuracy is still employed even if a range of different tool-use scenarios are used [Chen *et al.*, 2025].

LLM-as-a-judge, in which LLM outputs are evaluated by one or more other LLMs, is a popular method of evaluating LLM reasoning more generally. A range of prompting methods demonstrates correlation with human judgements [He *et al.*, 2024], localisation and correction of inconsistencies in reasoning chains can be performed [Tyen *et al.*, 2024], while debate can probe reasoning strength [Wang *et al.*, 2023a]. Alternatively, custom reasoning metrics may be defined to reflect the quality of reasoning undertaken [Hao *et al.*, 2024]; [Prasad *et al.*, 2023] score reasoning chains on correctiveness and informativeness, while [Golovneva *et al.*, 2023] define a range of custom reasoning quality metrics and train a model using synthetic- and human-labelled judgements.

Design of good explanations has been extensively described [Miller, 2019; Mohseni *et al.*, 2021; Hoffman *et al.*, 2019] *inter alia*. While this area of explainable AI (XAI) was developed alongside systems which typically produce a single decision or prediction, systems which explain technical or structured processes still benefit from the same design principles. Examples include prompting LLMs to generate baseline explanations of biomedical literature, augmented with knowledge retrieval to create lay-explanations [Guo *et al.*, 2024]; constructing explanations incrementally from a reasoning graph [Nuamah and Bundy, 2020]; and integrate prompting of LLMs and information retrieval for iterative refinement of natural language explanations [Quan *et al.*, 2025].

3 Tool Call Dependency Graphs

In this section, we outline our methodology for evaluating the meta-level reasoning of LLMs, shown in parts 1-4 of figure 1. First, we describe the scenario which we use to develop our graph generation algorithm: outputs from LLMs evaluated on a multi-hop dataset requiring arithmetic reasoning over numeric tabular data. We then develop an algorithm to create a directed acyclic graph from the list of tool call messages output by the LLMs. These graphs, which we refer to as Tool Call Dependency Graphs (TCDGs), are created post-hoc using only lists of messages generated by the LLM. Each node on a TCDG represents a tool call, and each edge $u \rightarrow v$ highlights where the result of the tool call u is predicted to have been used as an argument in tool call v. This captures about information about the flow of data throughout the reasoning process, and creates a structured representation of the solution as a whole. We then define three graph-theoretic metrics which take a TCDG as input, and output values which give tangible impressions of meta-level reasoning-related criteria.

Our TCDG is a tool for analysing the outputs of LLMs post-hoc, and not an assistive workflow, plan generation method, or chain-of-thought-based reasoning aid designed to support LLMs at inference time.

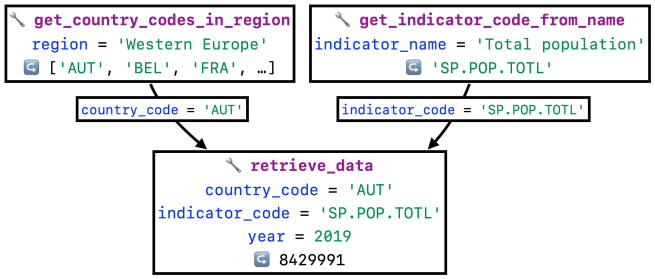

Figure 2: Illustration of a subgraph of a TCDG generated using our algorithm. The original question was *"What was the total population of Western Europe in 2019?"*. The LLM, as described in section 3.1 and figure 1, uses a set of tools to perform object-level reasoning. Here, two tool calls are performed which access data relating to country identification in a given region, and identifying a relevant indicator. A subsequent call uses the results of these calls as arguments, with labelled edges showing the flow of information.

3.1 Source Material and Problem Context

The dataset which we use to develop our graph construction algorithm is derived from [Ferguson *et al.*, 2026]. The authors evaluate several LLMs on a multi-hop question answering task requiring meta-level reasoning to create a plan to answer questions, and object-level reasoning to implement each step in the plan using arithmetic and tabular data retrieval. Question types are templated and are paired with gold standard answers against which the outputs of LLMs can be evaluated using final answer accuracy. To answer questions, object-level processes are implemented as elementary data retrieval and arithmetic tools which the LLM may call. Models are held in a loop in which models they are able to call tools and receive results before acting on the next step of the plan. This stage repeats, with the model making tool calls and receiving results, until a dedicated 'final answer' tool is called. As a test domain, questions are constructed around the values of World Bank indicator data across different countries and years [1].

However, to produce an answer, the LLMs may be required to perform upwards of 50 tool calls to obtain the necessary data and perform the required arithmetic. Therefore, evaluation of the system's meta-level reasoning ability using final answer accuracy alone cannot capture information about the actions taken in the intermediate process. This necessitates an alternative approach, embodied in our graph-theoretic solution which we describe below.

3.2 Algorithm

We create TCDGs using a two-stage algorithm. Inputs to the algorithm are an ordered list of tool call messages output from evaluation of LLMs on the above task. These are denoted $C = [c_1, \ldots, c_k]$, where $c_i = (t_i, a_i, r_i)$, and t_i is the ith tool call; a_i is a tool argument-value mapping; and r_i is the result of the call. The algorithm outputs a directed acyclic graph $G(N, E)$, where N is the set of nodes and E the edges. While we evaluate on the above example domain, we argue that our algorithm generalises to other instances of tool-based QA which require multiple steps of numeric reasoning and data retrieval because the algorithm requires only the sequence of tool calls to produce the TCDG. We do acknowledge a core limitation of our current implementation and algorithm in that we are unable to model QA processes which contain natural language reasoning or non-tool-based object-level reasoning. While tool call-oriented evaluation is a strength of our work given the interpretability of tools, at the moment we are unable to address free-text reasoning chains.

The first pass of the algorithm iterates through the tool call sequence and creates nodes for each call, as well as registering the values r_i output by the tool, which we term the provenance mapping, P. r_i may be a single element, or a list of values: if it is a list, we record each element separately in P. Stage 2 creates edges between nodes. For each argument value in each tool call node v, the most recent node u which produced this value is looked up using the provenance map. Then, if u is before the current node, an edge $u \rightarrow v$ is created. Each edge is labelled with the argument name and value which is produced by node u and used by node v . We are then left with a set of nodes representing each tool call, containing tool names, argument mappings, and results as node properties. These nodes are linked by edges which indicate where a value in the result of the tool call was consumed as an argument by a subsequent tool call, enabling explicit tracking of the information input to, and output by each tool call.

3.3 Deriving Metrics

Given the TCDG $G(N, E)$, we define three metrics which allow us to infer the quality of meta-level reasoning demonstrated by the LLM over the QA process. These metrics are designed to be generalisable to any tool-based QA setup where the object-level reasoning is undertaken entirely through a series of tool calls. Metrics that derive from the TCDG are not limited to our proposed metrics. Other graph-theoretic metrics may be defined to suit different tasks, for example, quantifying the number of exploratory reasoning chains which did not contribute to the answer.

When evaluating a model's performance across a dataset, the resulting metrics may be averaged across examples to report values to give a coarse-grained impression of reasoning performance. However, the strength of these metrics is that they reveal intricacies about the performance of models on single examples. Critically, the TCDG enables localisation of potential sources of error in individual examples, which would not be possible with the list of tool calls alone.

Final Answer Subgraph Ratio

Final Answer Subgraph Ratio, U, illustrates the efficiency of the model in terms of the proportion of tool calls which were explicitly used in the construction of the final answer. This gives an impression of the proportion of tool calls which 'mattered', and in consequence, the proportion which ultimately did not explicitly contribute to the final answer. Reasoning models may sometimes undertake alternative reasoning strategies [Yao *et al.*, 2023; Wang *et al.*, 2023b], which do not ultimately lead to a final answer. While small numbers of exploratory tool calls are expected, lowering U by a nominal amount, substantial proportions of the final TCDG dedicated

[1] https://data.worldbank.org/indicator

to exploratory strategies indicates weaker meta-level reasoning compared to pursuing a correct initial strategy.

Definition 1. *Let $n^\star$ refer to the node which provides the final answer, anc(n) be the set of nodes which are ancestors to a node n, and $|N|$ be the number of nodes in the graph. The **final answer subgraph ratio**, U_{final} is then given by*

$$U_{final} = \frac{|anc(n^\star) \cup \{n^\star\}|}{|N|} \tag{1}$$

High values are good, indicating that most tool calls explicitly contributed to the final answer. This indicates strong meta-level reasoning, showing that the model followed a process highly-informed by its previous steps, minimal use of implicit object-level reasoning in place of explicit tool use, and little deliberation of the correct strategy. Low values indicate that the outputs of most tool calls are not directly informing the final answer, showing poor meta-level reasoning by virtue of improper utilisation of the tools available to the model. This indicates that few explicit tool calls contributed to the object-level reasoning, suggesting that object-level processes were undertaken implicitly by the LLM (e.g., arithmetic). This diminishes trustworthiness, requiring further validation that those operations were correct.

Redundant Identical Tool Call Ratio

Second, we propose the *Redundant Identical Tool Call Ratio*, R, which quantifies the proportion of tool calls that were unnecessarily repeated. In our evaluated setup, tool APIs are static, which is made clear in their docstrings, and so models should understand that identical tool calls serve no benefit. While alternative implementations of tool-based QA approach may utilise tools which retrieve live information, and therefore whose results may change between calls, strong meta-level reasoning would be still be required and demonstrated by use of such tools at appropriate times during the QA process, so a notion of improper timing remains.

Definition 2. *Two calls (n_i, n_j) are identical if $(t_i, a_i) = (t_j, a_j)$. The number of occurrences of a given tool call is therefore $S(t, a) = \{i | (t_i, a_i) = (t, a)\}$. Since we want to count repeats of tool calls, with only one tool call being considered valid, the number of redundant calls is $|S(t, a)| - 1$. The **redundant identical call ratio** is then*

$$R = \frac{\sum_{(t,a)} \max(|S(t, a)| - 1, 0)}{|N|} \tag{2}$$

High values are worse, indicating a large number of repeated tool calls. This demonstrates poorer meta-level reasoning by indicating a lack of awareness of APIs interacted with, or tracking of the state of the process. Low values are good, indicating that the model understands the relevance and significance of steps previously taken, and comfortably builds upon them to perform the next stage of the process.

Provenance Completeness

Our third proposed metric is *Provenance Completeness*. This quantifies proportion of argument values which are grounded in earlier tool call results. In addition to the list of messages, this metric requires a tool schema (i.e., tool names and function signatures.) to infer where edges should exist, but do not. Factors that lower provenance completeness include the LLM performing steps without using tools, but in its standard text generation output. These may include a simple addition, or attempted recalling of memorised data. Regardless of any ability to perform addition without the aid of tools, models are instructed to use tools if there is a suitable tool available for a given task, and so any violation of this requirement is recorded negatively. However, the relative severity of this violation may be subject to interpretation by the end-user depending on the operation. An additional cause of lower provenance completeness is unnecessary rounding of results from arithmetic tool calls before using such values as arguments to subsequent tool calls causes breaks in provenance, which is detrimental to accuracy and trustworthiness.

Definition 3. *Let $\mathcal{V}(x)$ be an operator which returns the set of atomic values in a structure x. This will be used for returning the constituent values in the argument mapping a_i. We consider the set of all argument-value pairs for all tool calls:*

$$\mathcal{A} = \{(i, v) | v \in \mathcal{V}(a_i)\} \tag{3}$$

Each argument in this set may be either grounded in a previous tool output (that is, have an edge from n_i to n_j) or ungrounded. Therefore, a node is used if it has at least one successor: $|succ(n_i)| > 0$. An argument has provenance if it originated from a tool call:

$$(j, v) \in \mathcal{A}_{tool} \iff \exists (n_i, n_j) \in E \text{ such that } v \in \mathcal{V}(r_i) \tag{4}$$

*In other words, a value has provenance if there is an edge to it and that value is part of the results from the origin node's tool call. The **provenance completeness** is then given by*

$$C = \frac{|\mathcal{A}_{tool}|}{|\mathcal{A}|} \tag{5}$$

Provenance completeness relates to meta-level reasoning by indicating understanding of the relationship between different steps of the process, as well as demonstrating awareness of and proficiency using the available tools. High completeness indicates that most arguments are grounded and whose values can be trusted, indicating good meta-level reasoning, demonstrated by an understanding about how different stages of the process relate to each other, by virtue of the explicit use of tool results in subsequent calls, free of any prior rounding. Low completeness indicates that many arguments are ungrounded and not explicitly resulting from an earlier tool. This indicates poor meta-level reasoning by failing to grasp the importance of preserving information between steps of the process, and, depending on the circumstances, may result in lower accuracy and trustworthiness of the result of the process.

3.4 Results

Table 1 shows computed metrics for different model families and sizes on the QA task described in section 3.1 and [Ferguson *et al.*, 2026]. Our metrics reveal additional model-level performance insights, showing the potential of a TCDG platform in enabling deeper evaluation of meta-level reasoning, supporting hypothesis 1. Final answer subgraph ratio (U) show that high U is not an indicator of accuracy: while Llama

Model	Accuracy	U	R	C
Llama 3.3 70B Instruct	0.39	0.54±0.32	0.12±0.24	0.42±0.37
Qwen 3 4B	0.60	0.39±0.35	0.04±0.15	0.80±0.23
Qwen 3 14B	0.58	0.65±0.36	0.01±0.05	0.82±0.20
Qwen 3 32B	**0.84**	0.66±0.35	0.02±0.10	**0.86±0.14**
GPT 4.1 Mini	0.70	0.38±0.37	0.01±0.03	0.84±0.12
GPT 4o Mini	0.68	**0.77±0.24**	0.05±0.12	0.75±0.13
GPT 5 Nano	0.76	0.51±0.36	0.02±0.06	0.79±0.17
GPT 5 Mini	0.83	0.76±0.27	**0.01±0.02**	0.84±0.11

Table 1: Comparison of overall accuracy metrics to graph-derived metrics in section 3.3: final answer subgraph ratio U; redundant identical tool call ratio R; and provenance completeness C. Bold indicates the best performing model for that metric.

3.3 scored a low 39% accuracy, on average 54% of tool calls contributed to the answer. However, GPT 4.1 Mini's 70% accuracy was achieved with only 38% of tool calls explicitly contributing, suggesting a weaker understanding of the task and lack of continuity between reasoning steps.

Low identical tool call (R) values are widely reported, yet higher variances are found for Llama 3.3, Qwen 3 4B/32B, and GPT 4o mini; showing that even when overall accuracy is high, models may still make numerous unnecessary tool calls, indicating weaker understanding of the functionality of tools and relationship to the process as a whole.

Provenance completeness C appears generally high, but we report widespread *incompleteness*, with C values frequently showing that 15% of tool call arguments lacking grounding in any previous tool call. While the accuracy of, for example, Qwen 3 32B may be high, average C values of 86% raises questions about the fabrication of tool arguments, and therefore the reliability of such results. A single instance of missing provenance can be a critical weakness in the propagation of information: fabrication or early rounding of just one value can substantially alter the final answer, and so we must have a high threshold of acceptable C values.

4 TCDG-Augmented Explanations

We now build on our metrics by using provenance information derived from the TCDG to generate natural language explanations of the question answering process. This develops our above reflection that, while model-level statistics provide a coarse-grained overview, application to individual examples provides the greatest insight, particularly with respect to provenance completeness. We now demonstrate the ability of the TCDG to capture information about reasoning quality on an example-by-example level by incorporating such information into the generation of natural language explanations to describe the question-answering process.

4.1 Report Generation

Given the graph $G(N, E)$ created via the algorithm introduced in section 3.2, we iterate over all vertices and edges to create a YAML-formatted report describing the TCDG. The YAML format was chosen due to its structured format, while still facilitating brief natural language statements for comments on provenance-related issues. Additional virtues include its wide popularity, leading to ease-of-interpretation by LLMs. Markdown formatting was considered, but its structure as strong as that provided by YAML.

Top-level sections in the report are the nodes, edges, and what we refer to as *issues*. Each node's properties are provided, including tool name and result, then for each argument, its name, value, source node index and source tool name. The edges section shows the connections between nodes in terms of the origin node index, destination node index, and the value that the edge carries from the result of the origin node to the argument of the destination node. If any arguments in tool call nodes are not grounded, we collate these at the end of the report under the 'issues' heading, which summarises such information, highlighting cases where no origin node was found for a particular argument. Section B of the supplementary materials shows extracts from example nodes, edges, and issues sections.

4.2 Creating Explanations

Using the same list of tool call messages which are used to generate the TCDG in section 3.2, we prompt GPT-4.1 to create a natural language explanation of the process by which an answer was constructed. The prompt instructs a format of an opening sentence, five bullet points describing the process, and a closing summary sentence, encouraging the use of bold and italic text to draw attention to key aspects, breaking up the text into more digestible spans. This serves as our baseline.

We then prompt with further instruction and the YAML-formatted report, modifying the baseline with information in the report. We instruct GPT-4.1 to perform a minimal modification, inserting provenance-related issues where they are relevant to existing content in the explanation. This results in our 'TCDG-augmented' explanation. We explored the creation of entirely new explanations rather than the augmentation method described above. While this method integrates provenance information more fluidly, it results in substantially different surface forms to the augmentation method. With the user study in mind, to reduce the chance that users would happen to select a preference for one explanation over another purely based on a preference for the surface form, we proceeded with the augmentation method to control for this event and isolate the effect of our method. Prompts for both stages are shown in supplementary materials section C, with example explanations shown in section D.

4.3 User Study

To evaluate hypotheses 2, we ran an online human participation study in which we presented pairs of explanations to participants. Participants were sourced from Prolific[2] and the study built with Qualtrics[3]. Each pair consisted of one baseline explanation and its corresponding TCDG-augmented explanation, presented to users side by side titled 'Explanation A' and 'Explanation B'. The relative left/right positions of the baseline and TCDG-augmented explanations were randomised for each example, such that sometimes the baseline explanation appears on the left as explanation A, and sometimes on the right, as explanation B.

Participants were asked five questions about each pair (shown below) and instructed to indicate a preference for either explanation A, explanation B, or no preference. Participants were provided with annotation guidelines followed by a verification question confirming that the guidelines were understood. Guidelines were created to direct users to the relevant aspects of the explanation that we were evaluating, and instructed users not to consider aspects not under evaluation, such as the length of the explanation or the formatting. Participant questions were chosen to reflect key explainability criteria. These include focussing on whether the explanation has made it easier to understand if mistakes might have been made. Because trust is a difficult topic to evaluate, and given the potential for misplaced trust in an AI system, we avoid asking which explanation they trust the most. Instead, we ask about which explanation helps the user *decide how much* they trust the systems answer, that is, helps the user make a decision about whether to trust the answer.

Q1 *Which explanation most helps you understand if mistakes have been made?*

Q2 *Which explanation most helps you decide how much you trust the system's answer?*

Q3 *Which explanation most helps you make a judgement about whether the answer is likely to be correct?*

Q4 *Which explanation is the easiest to read and understand?*

Q5 *Overall, which explanation do you prefer?*

60 pairs of baseline/TCDG-augmented explanations were used in the study. 50 participants were recruited and shown 12 pairs each, resulting in 10 judgements per pair. No additional screeners other than English as a first language were applied to obtain a sample of the general population. To reflect the fact that explanations should be tailored to users' needs, future studies may examine the preference of certain types of explanation among different user groups, such as those with or without knowledge of AI systems or the domain at hand.

4.4 Results

We will now consider the extent to which we have support for hypothesis 2. To give an initial impression about the answers provided, figure 3 shows the proportion of votes for baseline and TCDG-augmented explanations across the five question

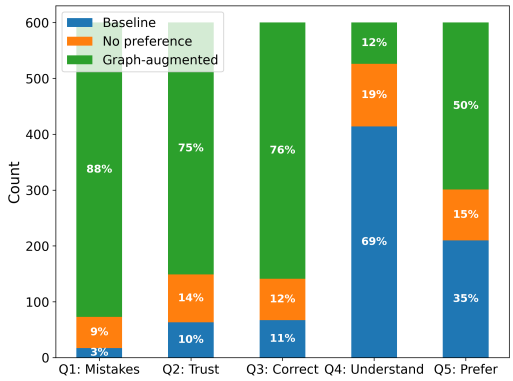

Figure 3: Proportion of votes for different response criteria.

blocks shown to participants. Preferences were not consistent across the different dimensions: for Q1-Q3, the majority of the votes were in favour of the TCDG-augmented explanations. However, for Q4, the majority vote was in favour of the baseline explanations, which tended to contain less information and had simpler surface forms. Q5 showed that opinion was more divided on overall preference, with approximately half of the votes for TCDG-augmented explanations.

Krippendorff's alpha [Krippendorff, 2018] which measures the extent to which participants gave the same ratings to the same items, adjusted for the agreement expected by chance. Across questions, alpha values range from 0.28 to 0.49, indicating modest agreement. As this is a subjective preference task, we interpret these values as reflecting genuine differences in how participants evaluated individual example pairs, rather than a lack of any underlying tendency.

TCDGs created from tool call sequences may exhibit a varying number of items in the *issues* section. In many cases, only one or two issues are present, whereas in others there may be over ten. For each issue in the TCDG, the baseline is modified to reference the issue. One question that arises from this is whether the number of issues in the TCDG, and therefore the number of statements which reference such issues in the explanation, affects preference towards either explanation. To assess whether the number of issues in each TCDG resulted in a stronger overall preference for baseline or TCDG-augmented explanations, we computed Spearman's rank for the number of issues and average preference per pair. Across Q1-Q5, we observed $-0.95 < \rho < 0.23$ with $p > 0.1$, indicating no systematic correlation between the number of issues and the preference towards either explanation. Given no evidence to suggest that the number of issues influences the preference towards either explanation, we use this as a basis to treat all pairs as exchangeable. We do not claim the absence of any underlying relationship, and further study may show that the judgements *are* impacted by the number of issues in the given examples, but we observed no such impact.

Finally, to test our core hypothesis that TCDG-augmented explanations are preferred by participants, we employed a one-sided exact binomial test. For the five blocks Q1-Q5, we summarised each participant's responses into a predominant response indicating which variant they preferred with

[2]https://www.prolific.com/
[3]https://www.qualtrics.com/

Question	TCDG	Base.	No pref.	TCDG pref.	95% CI	p
Q1: Mistakes	47	0	3	100.0%	[92.5%, 100.0%]	$\ll 0.01$
Q2: Trust	41	4	5	91.1%	[78.8%, 97.5%]	$\ll 0.01$
Q3: Correct	40	5	5	88.9%	[75.9%, 96.3%]	$\ll 0.01$
Q4: Underst.	4	40	6	9.1%	[2.5%, 21.7%]	1
Q5: Prefer	24	20	6	54.5%	[38.8%, 69.6%]	0.33

Table 2: Participant responses expressed as the number of participants that preferred a given explanation variant for each question block.

the highest frequency: TCDG-augmented, no preference, or baseline, shown in table 2 as TCDG, *Base.*, and *No pref.* respectively. TCDG *pref.* reports the proportion of participants favouring the TCDG-augmented explanation among participants with a directional preference, and we report a 95% confidence interval. As shown in the table by the high proportion of TCDG-preferring participants and the sufficiently small p-value, we have evidence to show that TCDG-augmented explanations were strongly and significantly favoured by participants across the criteria evaluated in Q1-Q3. However, on Q4, the question of which explanation is easiest to read and understand, we have evidence in the opposite direction: few participants indicate preference for the TCDG-augmented explanations, and preferences overwhelmingly favoured the shorter baseline explanations. While 54.4% of participants indicated overall preference for TCDG-augmented explanations, the confidence interval is wide and the resulting p-value of the binomial test indicates that we do not have evidence that TCDG-augmented explanations were preferred overall.

4.5 Discussion

TCDG-augmented explanations were strongly and significantly favoured by users for Q1-Q3, which concerned which explanation helped with understanding about mistakes that were made; deciding how much to trust the system; and judging whether an answer is likely to be correct. This provides a strong indication that information extracted from the TCDG is useful to users across these three key explainability criteria, showing that our TCDG-augmented explanations have strong utility for improving the explainability of answers containing long chains of tool-based reasoning.

However, when assessing which of the candidates were easiest to read and understand, participants expressed a strong preference for baseline explanations. We expect this to be due to explanation lengths: incorporating provenance information into TCDG-augmented explanations resulted in average length increase of 1.5× compared to the corresponding baseline. Given that the TCDG provides *additional* information over the baseline explanation, TCDG-augmented explanations are unavoidably longer. However, a consequence of this increased length is a reduction in overall ease of understanding, so a trade-off between information gain and brevity is vital.

Following on from this, opinion was more divided regarding the final 'overall preference' question. A weak, yet significant overall preference for TCDG-augmented explanations was observed, with approximately half of examples resulting in a majority vote for TCDG-augmented explanations. However, annotation of a substantial proportion of example pairs resulted in a majority vote for the baseline explanations. Evidence from the previous questions Q1-4 suggests that a possible reason for the baseline preference may be that, on balance, participants felt the additional information included in the TCDG-augmented explanation was not worth the trade-off in readability. Since participants overwhelmingly preferred the TCDG-augmented explanations along the above criteria, yet found them harder to read and understand, a trade-off is vital. We hypothesise that significant utility will be found in a graphical, interactive component to our explanations, and that a pure text medium is suboptimal for consuming the information contained in the explanation.

5 Conclusion and Future Work

In this paper, we introduced the topic of evaluating the meta-level reasoning ability of LLMs, manifested in a multi-hop question-answering task requiring multiple steps of numeric reasoning and data retrieval. As a proxy for investigating meta-level reasoning ability, we utilise the tool-use paradigm as an interpretable intermediate representation of a natural language plan of the question-answering process. After extracting the list of tool calls made by models when answering the question, we develop an algorithm to create a directed acyclic graph modelling the calls, which we term a Tool Call Dependency Graph (TCDG). This captures how tool calls, represented as nodes, utilise results from previous tool calls as indicated by edges showing the flow of information. From this TCDG, we propose three graph-theoretic metrics, and show that they provide tangible insight into the meta-level reasoning ability beyond final answer accuracy. Following this, we utilise the information contained in the TCDG to modify a baseline explanation created solely from the list of tool call messages. We then run a user study and show that our TCDG-augmented explanations offer significant utility over the baseline. However, we note that a text-only medium is suboptimal for delivering such information. Future work on an interactive, graphical medium will realise the potential of the graph-theoretic approach as a real-world solution to monitoring and oversight of LLMs applied to complex, numeric- and data-oriented tasks. Additionally, the ability of the TCDG to inform the LLM of potential mistakes or faulty reasoning at inference time may facilitate correction of faulty reasoning, improving the trustworthiness of answers.

Acknowledgements

This work was supported in part by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1) and the University of Edinburgh, School of Informatics and School of Philosophy, Psychology & Language Sciences.

References

[Abdin *et al.*, 2024] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 Technical Report, December 2024. arXiv:2412.08905 [cs].

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[Bundy, 1983] Alan Bundy. *The Computer Modelling of Mathematical Reasoning*. Academic Press, 1983.

[Chen *et al.*, 2025] Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Yuefeng Huang, Xiangcheng Liu, Wang Xinzhi, and Wu Liu. ACEBench: A Comprehensive Evaluation of LLM Tool Usage. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12970–12998, Suzhou, China, 2025. Association for Computational Linguistics.

[Cobbe *et al.*, 2021] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, November 2021.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[Ferguson *et al.*, 2026] Nick Ferguson, Alan Bundy, and Kwabena Nuamah. Exploring the Meta-level Reasoning of Large Language Models via a Tool-based Multi-hop Tabular Question Answering Task, January 2026.

[Gao *et al.*, 2023] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR, July 2023.

[Gemini Team, 2025] Gemini Team. Gemini: A Family of Highly Capable Multimodal Models, May 2025.

[Geva *et al.*, 2021] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. *Did Aristotle Use a Laptop?* A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, April 2021.

[Golovneva *et al.*, 2023] Olga Golovneva, Moya Peng Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023.

[Guo *et al.*, 2024] Yue Guo, Wei Qiu, Gondy Leroy, Sheng Wang, and Trevor Cohen. Retrieval augmentation of large language models for lay language generation. *Journal of Biomedical Informatics*, 149:104580, January 2024.

[Hao *et al.*, 2024] Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, Zhen Wang, and Zhiting Hu. LLM reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

[He *et al.*, 2024] Hangfeng He, Hongming Zhang, and Dan Roth. SocREval: Large Language Models with the Socratic Method for Reference-free Reasoning Evaluation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2736–2764, Mexico City, Mexico, 2024. Association for Computational Linguistics.

[Hoffman *et al.*, 2019] Robert R. Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. Metrics for Explainable AI: Challenges and Prospects, February 2019.

[Krippendorff, 2018] Klaus Krippendorff. *Content Analysis: An Introduction to Its Methodology*. Sage publications, 2018.

[Li *et al.*, 2022] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien De Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando De Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, December 2022.

[Llama 3 Team, 2024] Llama 3 Team. The Llama 3 Herd of Models, August 2024. arXiv:2407.21783 [cs].

[Lu *et al.*, 2025] Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A Stateful, Conversational, Interactive Evaluation Benchmark for LLM Tool Use Capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183, Albuquerque, New Mexico, 2025. Association for Computational Linguistics.

[Mialon *et al.*, 2023] Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: A survey. *Transactions on Machine Learning Research*, 2023.

[Miller, 2019] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, February 2019.

[Mohseni *et al.*, 2021] Sina Mohseni, Niloofar Zarei, and Eric D. Ragan. A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems. *ACM Transactions on Interactive Intelligent Systems*, 11:1–45, December 2021.

[Nuamah and Bundy, 2020] Kwabena Nuamah and Alan Bundy. Explainable Inference in the FRANK Query Answering System. In *European Conference on Artificial Intelligence (ECAI)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2441–2448, Spain, 2020. IOS Press.

[Parisi *et al.*, 2022] Aaron Parisi, Yao Zhao, and Noah Fiedel. TALM: Tool Augmented Language Models, May 2022.

[Patil *et al.*, 2024] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 37. Curran Associates, Inc., 2024.

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

[Prasad *et al.*, 2023] Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. ReCEval: Evaluating Reasoning Chains via Correctness and Informativeness. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10066–10086, Singapore, 2023. Association for Computational Linguistics.

[Qin *et al.*, 2024] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024.

[Quan *et al.*, 2025] Xin Quan, Marco Valentino, Danilo Carvalho, Dhairya Dalal, and Andre Freitas. PEIRCE: Unifying Material and Formal Reasoning via LLM-Driven Neuro-Symbolic Refinement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 11–21, Vienna, Austria, 2025. Association for Computational Linguistics.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

[Schick *et al.*, 2023] Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Nips '23. Curran Associates Inc., 2023.

[Tyen *et al.*, 2024] Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13894–13908, Bangkok, Thailand and virtual meeting, 2024. Association for Computational Linguistics.

[Wang *et al.*, 2023a] Boshi Wang, Xiang Yue, and Huan Sun. Can ChatGPT Defend its Belief in Truth? Evaluating LLM Reasoning via Debate. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11865–11881, Singapore, 2023. Association for Computational Linguistics.

[Wang *et al.*, 2023b] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

[Yao *et al.*, 2023] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc., 2023.