

Exploring State Tracking Capabilities of Large Language Models

Kiamehr Rezaee , Jose Camacho-Collados and Mohammad Taher Pilehvar

Cardiff NLP, School of Computer Science and Informatics

{rezaeek, camachocolladosj, pilehvarmt}@cardiff.ac.uk

Abstract

Large Language Models (LLMs) have demonstrated impressive capabilities in solving complex tasks, including those requiring a certain level of reasoning. In this paper, we focus on state tracking, a problem where models need to keep track of the state governing a number of entities. To isolate the state tracking component from other factors, we propose a benchmark based on three well-defined state tracking tasks and analyse the performance of LLMs in different scenarios. The results indicate that the recent generation of LLMs (specifically, GPT-4 and Llama3) are capable of tracking state, especially when integrated with mechanisms such as Chain of Thought. However, models from the former generation, while understanding the task and being able to solve it at the initial stages, often fail at this task after a certain number of steps.

1 Introduction

State tracking refers to the process of maintaining and updating an internal representation of a system’s condition or status over time, based on a sequence of observations or inputs. This internal representation, or the “state”, encapsulates relevant information from past inputs that is necessary to predict future outputs or behaviors of the system. This capability comprises three essential components: an internal representation or model of the system or “world”, the capacity to comprehend the effects of actions and updates on this world, and some form of on-demand memory system that maintains the current and updated versions of the world state.

This phenomenon has been studied in Large Language Models (LLMs) based on the Transformer architecture (Vaswani et al., 2017), highlighting potential shortcomings of these systems in tracking the state of games such as Othello and chess (Li et al., 2023; Toshniwal et al., 2021). The possible reasons behind these flaws could be attributed

to two main factors: (1) the single-modality text-based pre-training objective, which may not provide a comprehensive model of the real world for the system or may offer insufficient training signals, thereby limiting the model’s ability to effectively integrate and utilize contextual information, and (2) inherent issues with the underlying Transformer architecture, which might be inadequate for tasks that require consistent state tracking.

In this context, theoretical investigations into the computational boundaries of the Transformer architecture have revealed limitations, particularly regarding their ability to express sequential computation and the challenge of tracking a coherent state through intermediate steps while solving a problem (Merrill and Sabharwal, 2023; Dehghani et al., 2018). However, Li et al. (2024) suggested that in theory, by incorporating Chain of Thought, transformers can overcome their limitations and solve problems that are inherently serial and not parallelized (e.g. most state tracking tasks). One shortcoming of these theoretical studies is that they have not examined such limitations in the context of LLMs trained on natural language, leaving unanswered the open question of whether LLMs can acquire the capability of state tracking through training on general natural language data.

The main research question in this paper is whether state-of-the-art Transformer-based LLMs, trained on natural language textual data, can effectively manage state tracking. To this end, we provide a comprehensive assessment of the ability of LLMs to track states by designing a set of probing tasks. These tasks were designed to offer deeper insights into the capabilities and limitations of LLMs in maintaining consistent states throughout interactions. To limit confound effects, we have presented the tasks in a simple form along with baselines showing that LLMs are able to understand the tasks, so as to isolate the effect of state tracking capabilities. We diversified the datasets by

incorporating various configurations and designed the tasks to be as simple as possible for human solvers.

Our main contributions are threefold: (1) We propose a simple process to construct datasets for three carefully designed elementary tasks to highlight LLMs’ competence in state tracking, which can be used both as a benchmark and as a sanity check when deploying LLMs¹; (2) We perform a set of detailed experiments across LLMs to measure their performance in state tracking, isolating it from other factors; and (3) We provide an in-depth analysis to shed light on the possible factors impacting the performance of various systems.

2 Related Work

2.1 Theoretical Perspective

From an algebraic formal language theory perspective, previous studies have laid the theoretical foundation by forming connection between the problem of state tracking and the well-studied mathematical field of group theory (Liu et al., 2023; Merrill et al., 2024). This connection provides a way to establish the computational complexity class in which different state tracking problems reside, and a theoretical upper-bound to the complexity class of problems that are solvable by Transformer architecture has been provided (Merrill and Sabharwal, 2023). These studies suggest that tasks that are inherently “non-parallelizable” in nature and involve some sort of recurrence mechanism in their solution (e.g. most of the state tracking problems with the need for memory-based storage of intermediate states during repeated update applications), are “hard” for the Transformer architecture to solve.

The work by (Li et al., 2024) further extends these theoretical studies by suggesting that Transformer based models, when allowed to generate as many intermediate reasoning tokens as needed, can utilize their input window as a form of memory, effectively making them powerful enough to solve such inherently serial tasks.

A key distinction of these works to ours is their focus on transformers trained with specially designed objectives tailored to specific tasks. In contrast, we explore transformers in the context of LLMs trained using the language modeling objective, and investigate whether these models naturally learn to utilize their input tape as a form of interme-

¹We will release the simulation code used to generate the task datasets, along with the datasets used in our experiments upon acceptance.

diate memory to track reasoning steps and states, when trained on natural language data.

2.2 Memory Integration

An essential ability required to solve state tracking problem is to have an internal memory capable of storing and updating internal states. A fundamental drawback of the models based on the Transformer architecture compared to naturally recurrent architectures like Recurrent Neural Networks (e.g. RNNs), is that the only form of inference-time memory accessible to Transformers is their limited input window, which they can write to, whereas in the case of RNNs, in theory, they can update their internal representation of state infinite times.

According to Dehghani et al. (2018), the Transformer architecture, can become Turing-complete (hence able to solve state tracking problems) simply by integrating some form of read-write memory.

The evidence of improvements when integrating memory with Transformers, has been reported in various studies (Zhong et al., 2022; Wu et al., 2022). This proposed memory components are mainly suggested as a way to improve performance on tasks that rely on retrieving relevant information from documents. However, this type of memory augmentation might not be as advantageous for state tracking problems as they are not necessarily well-equipped to effectively utilize it for problem-solving tasks that involve tracking and updating states sequentially in multiple steps.

Perhaps more inline with theoretical findings of Li et al. (2024), in the context of unmodified Transformer-based LLMs, Wei et al. (2022) propose a method, namely Chain of Thought (CoT), in which the language model is simply instructed to explain the intermediate steps of solving the problem at hand, effectively guiding the model to use its input window as some sort of temporary memory.

In this work, as we are interested to see whether the Transformer-based models, with no further modifications, are able to handle state tracking problems by utilizing their limited input tape, we will provide evaluations of systems prompted in this manner, although other type of memory integration could be tried.

2.3 Multi-step Reasoning and State Tracking Evaluation

Previous studies have aimed to introduce relevant probing tasks in order to assess the proficiency of LLMs in tracking the underlying state that governs entities and their interactions throughout conversa-

tions. In this context, a branch of studies explore the capabilities of language models in dialogue state tracking (Williams et al., 2013; Budzianowski et al., 2018). These studies typically assess how well systems respond to queries based on the context provided in the chat history. Our work, however, differs in a key aspect: we investigate whether models can answer queries about state and information not explicitly present in the dialogue history.

Another branch of studies focuses on the ability of models to perform multi-hop reasoning, that involves multiple intermediate steps of inference (Yang et al., 2018; Welbl et al., 2018; Khot et al., 2020). These studies differ from ours as they assess models in terms of the ability to integrate information from diverse contexts and sources, which involves various forms of reasoning, such as logical and commonsense reasoning. In contrast, our work primarily involves procedural and sequential reasoning, where the task is to accurately apply a series of update statement changes. This allows for a detailed analysis of the model’s capability by introducing a controlled experimental environment.

Perhaps more aligned with the state tracking capability we are interested in is the `chess_state_tracking`² task in the Big-bench benchmark suite (Srivastava et al., 2023) could be mentioned. As the names suggest, the task is querying model about the final state of a chess board after a series of updates. However, this task does not fully capture what we are trying to measure, as the update statements are presented in an algebraic chess notation, which abstracts away the nuances of natural language.

Moreover, `tracking_shuffled_objects`³ task in Big-bench describes a set of objects that are held by a number of individuals and are swapped in a series of pairwise trades, and the model is asked about the final placement of an object. However, a notable challenge arises from the number of the update statements presented in this task, which lacks the necessary diversity to support comprehensive analysis.

3 Task Design

In this section, we outline the design perspective we applied throughout the process of creating the set of simple tasks. Our goals are two-fold: (1)

²The dataset can be found at https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/chess_state_tracking

³Dataset at https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/tracking_shuffled_objects

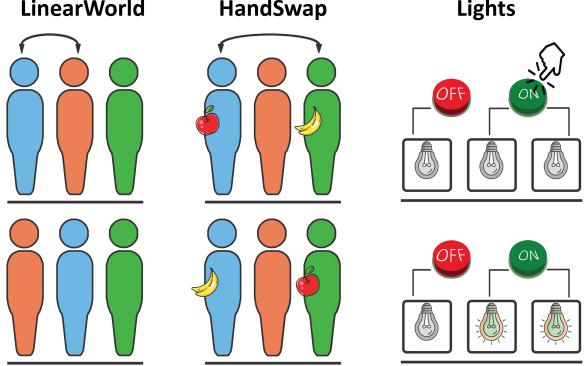


Figure 1: A simple illustration of the initial state of the tasks (top) and the updated state after a singular update step (bottom) of the tasks. Note that the configuration shown for Lights is not complete as only two switches are depicted in the image; for the exact configuration of rooms and switches, please refer to section 4.2.

first, to keep the tasks simple enough for humans to solve, and (2) second, to provide different scenarios where the ability to track some form of internal state is essential to solving the given task. Aggregating the results of different tasks with various wordings also helps to isolate the state tracking aspects we are interested in and eliminate the variance in performance caused by several factors such as the surface form (the wording) of the prompts designed for each task.

Each task in this set consists of an initial state of an auxiliary world (containing entities and/or objects) presented to the model, followed by a series of elementary updates to the aforementioned world, and finally a query about the final state of the world. We use the term "depth" to refer to the number of world updates provided to the model before presenting the query. In other words, the depth indicates the recurrence required for the model to apply the transition function from the initial state to obtain the final state.

We present three different tasks, referred to as **LinearWorld**, **HandSwap**, and **Lights**. The first two are different representations of the same underlying problem, which involves tracking different permutation configurations of a number of objects. An illustration of these three tasks is presented at Figure 1. We explain each task separately in the following subsections.

3.1 LinearWorld

The first task in this suite of tasks consists of an auxiliary linear world where a number of entities are standing on a line in a specific order. The initial configuration of this world is presented to the

model by the name and position of entities, with an integer indicating their position (e.g. "entity A is standing at point 3") relative to the origin at coordinate (0). The state of the world changes whenever two entities swap their positions (e.g., 'Entity A swaps positions with Entity B'). Finally, the query in this setting, is simply a choice from list of possible relative position of two arbitrary entities (e.g. "Where is entity A standing with respect to entity C now?" with options being ["right", "left"]). An example of the prompt we use for this task in provided in Appendix A.1.

3.2 HandSwap

Another form of the same underlying problem as the LinearWorld task (which deals with permutation combinations of a number of entities) is where the update statements address indices rather than the entities themselves. For example, in case of the LinearWorld task, an update might be given as "whoever is standing in position 3 changes positions with whoever is standing in position 1," rather than "entity A swaps positions with entity B." This task is an extension to the *tracking_shuffled_objects* task in the Big-bench suite (Srivastava et al., 2023), in which we present more depth (number of update statements) to the instances.

As for this task, the auxiliary world consist of a number of individuals holding various items in their hands. The initial configuration of this world is described by the name of entities, along with the name of the item they are holding at the moment. Each update involves two individuals in the world exchanging the item in their hands ("e.g. "entity A swaps hands with entity B"). The query involves selecting an item from a list of potential objects that an arbitrary individual of choice might be holding (e.g. "What is entity A holding in their hands right now?" with options such as ["apple", "banana", "nothing"]). An example of the prompt we use for this task in provided in Appendix A.2.

Note that here, the subjects of the updates are the holdable items, while the entities act as the containers or indices. Although the underlying problem remains the same as the LinearWorld problem, by introducing variety through different wordings compared to the LinearWorld task, we aim to minimize the impact of surface form bias in our evaluations.

3.3 Lights

The final task involves a fundamentally different problem compared to the other two. In this task, the

model of world contains a number of rooms each having a light bulb in them. The auxiliary world also contains a number of individual switches. The initial configuration of this world is defined by the status of each light in each room (e.g. "The light in room A is initially off"), and how the functionality of each switch affects the light in each room (e.g. "When switch 1 is pressed, it turns the light of room A off"). The updates here are simply represented by a series of key presses (e.g. "The switch 4 is pressed"), and the query in this configuration asks for the final status of the light in a room of choice (e.g. "What is the final light status in room C?" with options being ["off", "on"]). An example of the prompt we use for this task in provided in Appendix A.3.

4 Experimental Setting

In this section, we describe the various experimental configurations used throughout our experiments.

4.1 Comparison Systems

Our main group of evaluation systems consists of Transformer-based LLMs, specifically GPT-3.5-turbo, GPT-4o, Mixtral-8x7B-Instruct (Jiang et al., 2024), LLama3-8B and LLama3-70B⁴. We evaluate these models via a prompt-based approach. Each prompt consists of an instruction statement that sets the model up for the task at hand, a natural language representation of the initial state, a series of update statements, and finally, a query presented in natural language form.

In addition, to enhance model performance, we provide a solved example to the model in advance. Moreover, since the input tape of transformers can function as a form of on-demand memory for solving tasks, we conducted a variation of experiments with Chain of Thought (CoT) instructed models. Specifically, we instructed the models to include their reasoning process in the output to assess how impactful this type of memory is for task solving. Example of the prompts used for each task can be found in Appendix A.

4.2 Task Configurations

Regarding the number of entities in the main configuration of the LinearWorld task, we opt for three individuals in this world to simplify the task

⁴At the time this paper was written, there were no official papers available for GPT-4o or LLama3 models. Please refer to the respective blog posts at the following addresses: GPT-4o: <https://openai.com/index/hello-gpt-4o/>, LLama3: <https://llama.meta.com/llama3/>.

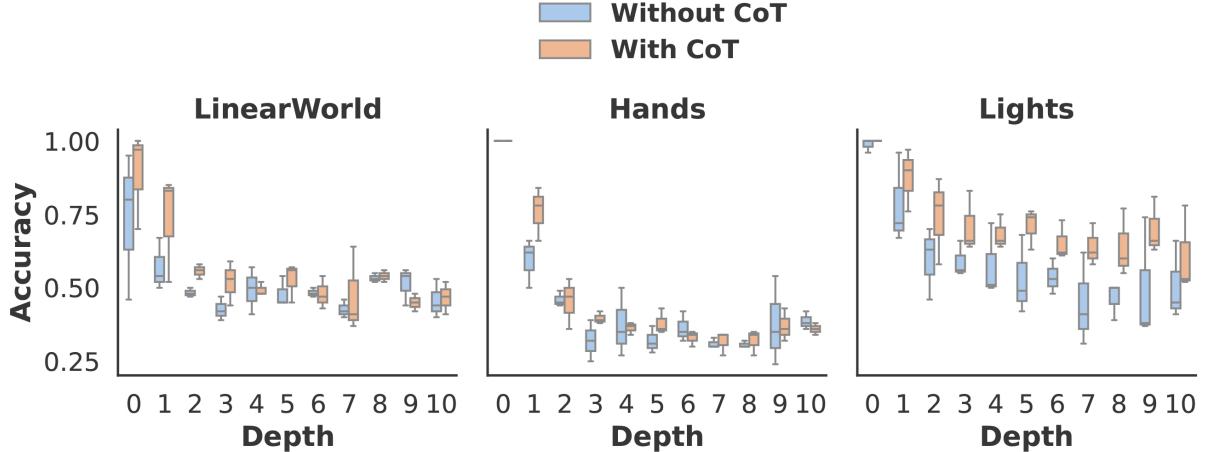


Figure 2: Average accuracy at different depths across tasks (left: LinearWorld, middle: Hands, and right: Lights) for all systems except for the two top performers which use Chain of Thought (CoT), i.e., Llama3 70B and GPT-4.

for Transformers. However, we also conducted complementary experiments with various configurations of this task, including experiments with five entities, as discussed in section 6. As for **HandSwap** task, similar to **LinearWorld**, the world consist of three individuals. Moreover, for the number of objects, we opt for three holdable options: "apple," "banana," and "nothing," which are swapped between the three entities. In the **Lights** task, we opt for three rooms and five switches. Each of the switches has different functionalities, as illustrated in the example prompt in Appendix A.3. These numbers are selected to ensure a manageable yet challenging range of possible states.

For each task, we developed a simulator to model the target world. Starting from an initial state, we sampled 50 episodes for each experiment, where each episode represents a trajectory from the initial state presented to the model to the final state after a sequence of 10 update statements, along with the corresponding intermediate states that occurred during these updates. The model is presented to these episodes in a prompt template format. The generation temperature parameter is set to ensure that the model’s responses are deterministic. Additionally, we apply minimal regex processing to the model’s output to extract the final label. To evaluate performance, we computed the accuracy of the models’ answers to queries at each depth of the update sequences across all the episodes.

5 Results

Table 1 illustrates the results of the evaluations. In what follows, we discuss some of the observations we have made:

Impact of Depth. The first observation is the ef-

fect of state tracking depth on system performance. The results indicate two distinct behaviors: smaller models (such as the LLama3 8B variant) and older models (GPT-3.5 and Mixtral) experience a significant drop in performance once the depth reaches 2, in contrast to larger and newer models (LLama3 70B variant and GPT-4o) which maintain acceptable accuracy even at greater depths, particularly when combined with CoT.

Interestingly, almost all LLMs (with the exception of LLama3 8B) have near-perfect accuracy at depth 0 over all the tasks, where no update statements are present. This suggests that the models can comprehend the natural language version of the task provided in prompts. They only struggle when it comes to following the update statements, due to the lack of access to on-demand memory.

Impact of Chain of Thought. Another evident observation is the benefit of incorporating Chain of Thought (CoT) into the model prompting process. With CoT, systems can use the input context as a form of memory to store intermediate states, thereby making it easier to solve the tasks at hand and follow the intermediate states of the evolving underlying environment.

Figure 2 shows the difference in the performance averaged across all models, excluding the two best-performing ones (LLama3 70B and GPT-4), when integrated with CoT compared to when CoT is not employed. The trend clearly highlights the benefit of incorporating CoT into the process.

Stateful Vs Stateless. One possible question is the extent to which LLMs rely on the initial state provided to them, potentially disregarding any update statements and solely using the initial state to answer the query. To this end, we designed a con-

Task	System	Depth										
		0	1	2	3	4	5	6	7	8	9	10
Linear	GPT 3.5	0.95	0.54	0.48	0.39	0.41	0.45	0.50	0.40	0.53	0.54	0.53
	GPT 3.5 CoT	0.97	0.85	0.56	0.59	0.48	0.45	0.47	0.37	0.56	0.45	0.47
	Mixtral	0.80	0.67	0.47	0.47	0.57	0.45	0.47	0.42	0.55	0.44	0.40
	Mixtral CoT	1.00	0.83	0.53	0.53	0.48	0.57	0.43	0.41	0.54	0.48	0.41
	Llama3 8B	0.46	0.50	0.50	0.42	0.50	0.54	0.48	0.46	0.52	0.56	0.44
	Llama3 8B - CoT	0.70	0.52	0.58	0.44	0.52	0.56	0.54	0.64	0.52	0.42	0.52
	Llama3 70B	1.00	0.82	0.54	0.34	0.56	0.66	0.52	0.52	0.62	0.56	0.52
	Llama3 70B - CoT	0.96	0.86	0.88	0.80	0.78	0.90	0.82	0.86	0.76	0.72	0.84
	GPT 4o	0.90	1.00	1.00	1.00	0.96	1.00	1.00	1.00	1.00	0.98	1.00
	GPT 4o CoT	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	0.96
	Stateless	1.00	0.48	0.58	0.42	0.61	0.53	0.55	0.39	0.54	0.40	0.52
	MFC	0.54	0.52	0.53	0.51	0.52	0.53	0.55	0.53	0.52	0.54	0.56
Hands	GPT 3.5	1.00	0.62	0.45	0.25	0.27	0.31	0.35	0.33	0.30	0.35	0.38
	GPT 3.5 CoT	1.00	0.84	0.47	0.39	0.37	0.35	0.35	0.27	0.35	0.43	0.36
	Mixtral	1.00	0.66	0.49	0.39	0.35	0.37	0.32	0.30	0.30	0.24	0.42
	Mixtral CoT	1.00	0.78	0.53	0.38	0.38	0.43	0.34	0.34	0.27	0.32	0.38
	Llama3 8B	1.00	0.50	0.44	0.32	0.50	0.28	0.42	0.30	0.32	0.54	0.36
	Llama3 8B - CoT	1.00	0.66	0.36	0.42	0.34	0.36	0.30	0.34	0.34	0.36	0.34
	Llama3 70B	1.00	1.00	0.58	0.38	0.38	0.32	0.34	0.32	0.24	0.36	0.24
	Llama3 70B - CoT	1.00	1.00	0.72	0.86	0.90	0.82	0.84	0.82	0.82	0.60	0.62
	GPT 4o	1.00	1.00	0.94	1.00	1.00	1.00	1.00	1.00	1.00	0.90	1.00
	GPT 4o CoT	1.00	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Stateless	1.00	0.35	0.40	0.37	0.33	0.40	0.31	0.34	0.30	0.29	0.34
	MFC	0.37	0.41	0.41	0.37	0.37	0.37	0.38	0.35	0.38	0.40	0.43
Lights	GPT 3.5	0.96	0.67	0.46	0.56	0.51	0.42	0.53	0.41	0.50	0.38	0.45
	GPT 3.5 CoT	1.00	0.76	0.58	0.66	0.64	0.63	0.61	0.58	0.55	0.63	0.53
	Mixtral	1.00	0.72	0.63	0.55	0.50	0.49	0.48	0.31	0.39	0.37	0.41
	Mixtral CoT	1.00	0.97	0.87	0.83	0.75	0.76	0.73	0.72	0.77	0.81	0.78
	Llama3 8B	1.00	0.96	0.70	0.66	0.72	0.68	0.60	0.62	0.50	0.74	0.66
	Llama3 8B - CoT	1.00	0.90	0.78	0.64	0.66	0.74	0.62	0.62	0.60	0.66	0.52
	Llama3 70B	1.00	0.96	0.90	0.92	0.86	0.88	0.88	0.74	0.82	0.58	0.68
	Llama3 70B - CoT	1.00	1.00	0.98	0.98	1.00	0.98	0.96	0.96	0.98	1.00	0.98
	GPT 4o	1.00	0.96	0.98	1.00	1.00	1.00	0.98	1.00	1.00	1.00	1.00
	GPT 4o CoT	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98
	Stateless	1.00	0.52	0.56	0.53	0.50	0.47	0.49	0.39	0.53	0.61	0.44
	MFC	0.58	0.65	0.56	0.52	0.51	0.54	0.54	0.56	0.53	0.61	0.54

Table 1: The main evaluation results of systems on different tasks reported in terms of accuracy at various depths. **Linear** and **Hands** refer to LinearWorld and HandSwap tasks respectively. Also, systems with Chain-of-Thought integrated in their evaluation are indicated with "CoT" in their names.

trol stateless baseline, that is simply a system that disregards all update statements and answers each query solely based on the provided initial state.

The results illustrated in Table 1 suggest that although not perfect, LLMs are in general able to answer queries about the updated environment more accurately than a naive stateless system. This observation indicates that the models which fail at greater depths, still follow the state to some extent.

6 Analysis

We present additional control experiments and analyses to complement the main results presented above. The experiments in this section are conducted on the LinearWorld task due to its simplicity and flexibility to study alternative configurations.

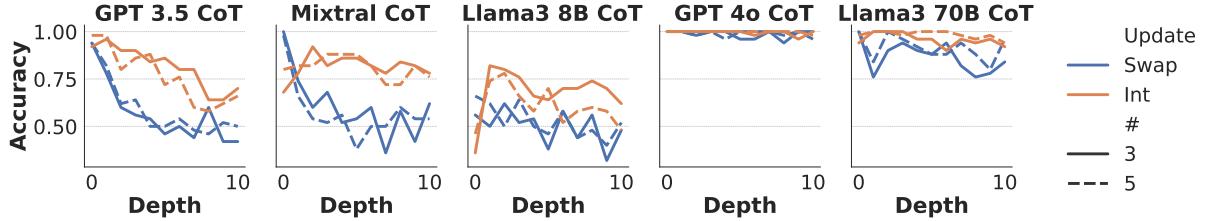


Figure 3: Accuracy at different depths comparing the "swap" update type with the "integer" update type in the LinearWorld task (the state-dependent query variants) for all CoT integrated systems.

6.1 Control Experiments

The control experiments and the main findings are summarized below. In the appendix (Tables 4 and 3), we include the full experimental results.

Number of Individuals. Increasing the number of entities (in this case people) in the configuration of LinearWorld task, adds to the complexity of the choice space for the model; the larger the number of entities gets, the more possible configurations exist in which they can stand in a line, thus increasing the number of possible states the model must track. With that in mind, we designed a version of the LinearWorld task with 5 entities standing in a line, instead of the original 3 entities.

Contrary to expectations, the models' performance does not drop when more entities are introduced to the problem. One possible explanation is that with more individuals, there are more possibilities to choose two random entities for the final query whose relative positions remain unaffected by the update statement. In that case, models could simply refer to the initial state of the entities to answer the query. We will see further evidence of this behavior when comparing the "state-dependent" and "random" query types, as discussed next.

State Dependant Queries. One consequence of having a greater number of individuals is that the chosen query more often asks about the relative position of two entities whose positions have not been modified by the updates. This, in turn makes the task easier for the model by simply answering the queries solely based on the initial state. In order to prevent that, we introduce a setting in which the person placement of each query is chosen so that the relative positions of entities in question differ from their initial state. The results for this setting, including a figure plotting the results by depth (figure 5), are shown in the appendix due to space constraints. In general, the results follow a similar trend in this and the default setting. However, the performance gap between query types is most pronounced in models that excel at lower depths but struggle at higher depths, with the default query

type (random) being the easier of the two.

Integer Displacement. In order to assess the impact of various update types, we introduce a variant of LinearWorld task, in which the update statements are simply the displacement of one of individuals by an arbitrary integer value (e.g. "Entity A moves 5 units to the left"). Following the update statements presented in this manner, requires a minimal mathematical calculation to determine the final positions of entities as a first step, and then deducing their relative positions to answer the query. One possibility is that models perform mathematical operations by simply memorizing the results of summing two small integers that have appeared in their pre-training data. To make this less plausible, we introduce initial state positions and displacements with large integer values as well, reducing the likelihood that these specific number combinations have occurred in the pre-training data.

As illustrated in Figure 3, systems surprisingly exhibit superior performance in the integer setting compared to swap. A subsequent qualitative analysis revealed that struggling systems fail to make logical statements and update the state coherently after receiving swap updates. In contrast, models tend to follow state changes more logically in the integer update setting. Some examples of errors can be found in the Appendix. A key difference between swap and integer displacement update types is revealed upon inspecting the responses: models attempt to isolate updates regarding each entity from all other updates, then aggregate and solve them out of sequence (e.g. $5 + 3 - 2 = 6$ as to solve the position of one person after three updates instead of performing each update separately).

6.2 Proficiency in Mathematical Operations

To assess how models perform when introduced to mathematical operations, we analyze their responses through a simple approach. We search for simple mathematical expressions in the model outputs using basic regex matching, and then we compute the percentage of correctly calculated mathe-

mathematical expressions by the model.

Figure 4 presents the accuracy of the model’s calculations as well as the average number of mathematical expressions written in the output response by the model per generation. GPT-4 demonstrates superior performance not only in mathematical calculations but also in generating a higher number of expressions per output. Moreover, the difference between the number of expressions in CoT versus non-CoT settings is less pronounced for GPT-4 (we will analyse this specific behaviour in the following section). In contrast, other models produce fewer expressions when CoT is not employed and generally exhibit lower accuracy in their calculations. This difference is particularly reflected in the results when comparing the performance of the large variant of Llama3 on integer-based tasks with and without CoT prompting.

Notably, the models exhibit decreased calculation accuracy when presented with large integer numbers compared to smaller ones. However, this deficiency is less marked in the performance of the models on the LinearWorld task. This may be attributed to the fact that in scenarios involving large integers, there is more room for calculation errors while still maintaining the correct relative order of individuals.

6.3 Response Length

As a basic proxy to measure how often models employ CoT or other sort of reasoning in their responses (whether instructed to do so or otherwise), we calculate Pearson’s correlation coefficient between the model response length and the depth of the query. A larger correlation indicates that the system is engaging more in deeper queries, suggesting a higher likelihood of a reasoning process being articulated by the model.

The results reported in Table 2 suggest that better-performing models demonstrate a stronger correlation between response length and the depth at which the response is generated. This indicates a higher likelihood of Chain of Thought reasoning being activated in these systems. As expected, the CoT mechanism is more likely to be activated in systems that are explicitly instructed to employ it.

Interestingly, a comparison of CoT activation across different systems reveals that for GPT-4, the CoT mechanism is activated to a similar degree regardless of explicit instructions. This suggests that GPT-4 employs CoT reasoning intrinsically, even when not specifically prompted to articulate its reasoning process in the output. These findings

align with the observation that GPT is more likely to write mathematical expressions in the tasks with integer-based update type, even without explicit instructions regarding the activation of CoT, as discussed in the previous section.

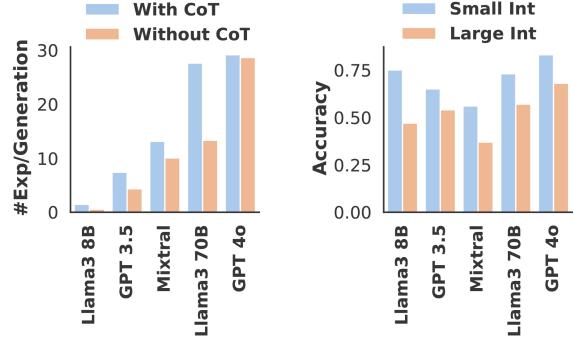


Figure 4: Average number of mathematical expressions per model response for all the models (left), and accuracy of generated expression evaluations across models integrated with CoT (right).

System	Pearson r
Llama3 8B / + CoT	0.01 / 0.12
GPT 3.5 / + CoT	0.03 / 0.04
Mixtral / + CoT	0.32 / 0.46
Llama3 70B / + CoT	0.35 / 0.79
GPT 4o / + CoT	0.85 / 0.91

Table 2: Pearson correlation between response length and generation depth across models.

7 Conclusion

In this paper, we investigated the problem of state tracking in LLMs, specifically their ability to maintain state through sequential and repetitive updates by conducting a set of controlled experiments. For this, we have constructed a new benchmark based on three related tasks that are specifically designed to test the state tracking capabilities of LLMs. Our experiments indicate that the newer generation of models, such as GPT-4 and Llama3 70B (the larger variant of the Llama3 models), can effectively maintain state by keeping intermediate states, especially when integrated with the chain of thought reasoning technique. In contrast, the smaller variant of the Llama3 model, Llama3 8B, and older generation LLMs, such as Mixtral and GPT-3.5, struggle to utilize their input context as a form of inference-time note-keeping memory for state tracking after a certain number of update statements.

Limitations

Despite the comprehensive nature of our study, several limitations must be acknowledged. First, our experiments were conducted on a limited set of models which may not fully capture the broader capabilities and nuances of state tracking across different architectures and sizes. Second, the prompts used in our experiments were also limited, which could affect the generalizability of our findings. Variations in prompt design might yield different results, highlighting the need for further exploration with diverse prompting strategies. Additionally, some of our analysis was confined to a single task, which may not provide a comprehensive understanding of state tracking across different contexts. Furthermore, we limited our experiments to a maximum depth of 10 update statements. This restriction might overlook the models’ performance in handling deeper sequences of updates.

References

- Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.
- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.
- Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lampe, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.
- Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.
- Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*.
- Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations*.
- Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023. Transformers learn shortcuts to automata.
- William Merrill, Jackson Petty, and Ashish Sabharwal. 2024. The illusion of state in state-space models. *ArXiv*, abs/2404.08819.
- William Merrill and Ashish Sabharwal. 2023. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 432 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615.
- Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. 2021. Chess as a testbed for language model state tracking. In *AAAI Conference on Artificial Intelligence*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.
- Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France. Association for Computational Linguistics.
- Yuhuai Wu, Markus Norman Rabe, DeLesley S. Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *ArXiv*, abs/2203.08913.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering.

In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Zexuan Zhong, Tao Lei, and Danqi Chen. 2022. *Training language models with memory augmentation*. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5657–5673, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Prompt Examples

In this section, we present examples of prompts used in our experiments. The examples provided here employ CoT. As for the simple non-CoT versions of these prompts, we simply omit the Reasoning parts, and the instructions are adjusted accordingly to not request the model’s reasoning process.

Note: The red italicized comments are not part of the actual input given to the model but serve to clarify various aspects of the prompts for the reader.

A.1 LinearWorld

– This is the instruction we provide for the model: –

Consider a 2D world with three individuals, each starting at a given initial position, and a series of movements for each person, where ‘left’ signifies a move towards negative coordinates and ‘right’ denotes a move towards positive coordinates. Your task is to calculate and report the final position of each person after the specified sequence of movements and respond to a given query about their respective positions. Your response should strictly follow this format: ‘Answer: [[response]]’, the response in the double-closed brackets is either left or right, e.g, ‘Answer: [[left]]’ and ‘Answer: [[right]]’. Here is an example:

– Here we provide an example for the model: –

Person A starts at position 0
Person B starts at position 1
Person C starts at position 2

- Person C swaps position with Person A.
- Person A swaps position with Person B.

Query: Where does Person B stand with respect to Person C now?

Options: left , right

Reasoning: Considering all the previous movements, Person B is at position 2. Person C is at position 0. Since 2 is greater than 0, Person B is to the right of Person C.

Answer: [[right]]

– Now we provide the actual test instance: –

Now your turn:
Person A starts at position 0
Person B starts at position 1
Person C starts at position 2
Person B swaps position with Person C.
Person C swaps position with Person A.
(Note that this instance has a depth of 2)
Query: Where does Person B stand with respect to Person A now?
Options: left , right
Reasoning:

A.2 HandSwap

– This is the instruction we provide for the model: –

Consider three individuals, each holding an object. Given initial object that each individual holds, and a series of swaps with other individuals, your task is to respond to a given query about the final object that one of the individuals is holding. Your response should strictly follow this format: ‘Answer: [[response]]’, with the response in the double-closed brackets chosen from this list of the options: [apple , banana , nothing], e.g, ‘Answer: [[apple]]’ or ‘Answer: [[banana]]’ or ‘Answer: [[nothing]]’. Here is an example:

– Here we provide an example for the model: –

Person A is initially holding apple
Person B is initially holding banana
Person C is initially holding nothing
Person B swaps hands with Person C.
Person C swaps hands with Person B.
Query: What is in the hands of Person B now?
Options: apple , banana , nothing
Reasoning: Considering all the previous swaps, Person B is now holding banana.
Answer: [[banana]]

– Now we provide the actual test instance: –

Now your turn:

Person A is initially holding apple

Person B is initially holding banana

Person C is initially holding nothing

Person C swaps hands with Person B.

Person A swaps hands with Person C.

Person C swaps hands with Person A.

(Note that this instance has a depth of 3)

Query: What is in the hands of Person C now?

Options: apple , banana , nothing

Reasoning:

A.3 Lights

– This is the instruction we provide for the model: –

Consider a scenario where there are a set of switches and multiple rooms each equipped with a light. Each switch affects the state of certain lights when pressed. Given the initial state of the lights in each room and the effects of pressing each switch, your

task is to respond to queries about the final state of a specific room after a sequence of switch presses. Your response format must strictly follow the following pattern: ‘Answer: [[response]]’, where the response enclosed in double brackets indicates whether the light is on or off, for instance, ‘Answer: [[on]]’ or ‘Answer: [[off]]’. Here is an example:

– Here we provide an example for the model: –

The light in room A is initially off.
The light in room B is initially off.
The light in room C is initially off.
When switch 1 is pressed, it turns the lights of room A on.
When switch 1 is pressed, it turns the lights of room B off.
When switch 2 is pressed, it turns the lights of room B on.
When switch 2 is pressed, it turns the lights of room C off.
When switch 3 is pressed, it turns the lights of room C on.
When switch 3 is pressed, it turns the lights of room A off.
When switch 4 is pressed, it turns the lights of room A, room B and room C on.
When switch 5 is pressed, it turns the lights of room A, room B and room C off.
The switch 3 is pressed.
The switch 3 is pressed.
Query: What is the final light status in room A?
Options: on, off
Reasoning: Starting from the first key press:
The switch 3 is pressed and it turns the light in room A off. The switch 3 is pressed and it turns the light in room A off. Therefore the final status of the light in room A is off.
Answer: [[off]]

– Here we provide the actual test instance: –

Now your turn:
The light in room A is initially off.
The light in room B is initially off.
The light in room C is initially off.
When switch 1 is pressed, it turns the lights of room A on.
When switch 1 is pressed, it turns the lights of room B off.
When switch 2 is pressed, it turns the lights of room B on.
When switch 2 is pressed, it turns the lights of room C off.
When switch 3 is pressed, it turns the lights of room C on.
When switch 3 is pressed, it turns the lights of room A off.
When switch 4 is pressed, it turns the lights of room A, room B and room C on.
When switch 5 is pressed, it turns the lights of room A, room B and room C off.
The switch 3 is pressed.
The switch 5 is pressed.
The switch 5 is pressed.
The switch 1 is pressed.

(Note that this instance has a depth of 4)

Query: What is the final light status in room C?
Options: on, off
Reasoning:

B Model Response Examples

In this section, we present examples of model responses in instances where errors occur. Note that a simplified version of the input to the model has been provided for each response. However, the models do not receive the input in this simplified format. For the full template in which these inputs are presented, please refer to section A.

Note: The red italicized comments are not part of the actual model responses but serve to clarify various errors of the systems for the reader.

B.1 Llama3 8B

Integer. Sample input:

A starts at 1.
B starts at 2.
C starts at 3.
B moves 2 to right.
C moves 5 to left.
A moves 5 to right.
B moves 5 to right.
A moves 9 to left.
C moves 3 to left.
A moves 15 to right.
B moves 8 to right.
A moves 20 to left.
Where does B stand with respect to C?

Sample response:

Person A starts at position 1 Person B starts at position 2 Person C starts at position 3
Person B moves 2 units to right.
-> Person B is now at position 4
Person C moves 5 units to left.
-> Person C is now at position -2
Person A moves 5 units to right.
-> Person A is now at position 6
Person B moves 5 units to right.
-> Person B is now at position 9
Person A moves 9 units to left.
-> Person A is now at position -3
Person C moves 3 units to left.
-> Person C is now at position 6 **Wrong!**
Person A moves 15 units to right.
-> Person A is now at position 12
Person B moves 8 units to right.
-> Person B is now at position 5 **Wrong!**
Person A moves 20 units to left.
-> Person A is now at position -8
Query: Where does Person B stand with respect to Person C now? Options: left, right Answer: right

Swap. Sample input:

A starts at 0.
B starts at 1.
C starts at 2.
C swaps with A.
A swaps with B.
B swaps with A.
B swaps with C.
B swaps with A.
C swaps with A.
C swaps with B.
C swaps with B.
Where does B stand with respect to C?

- starts at position 2
- moves 2 units to left: $2 - 2 = 0$
- moves 5 units to left: $0 - 5 = -5$
Person C:
- starts at position 3
- moves 8 units to right: $3 + 8 = 11$
- moves 3 units to right: $11 + 3 = 14$
- moves 16 units to left: $14 - 16 = -2$
- moves 20 units to right: $-2 + 20 = 18$
Now, let's compare the final positions of Person A and Person B: Person A is at position 19 Person B is at position -5 Since 19 is greater than -5, Person A is to the right of Person B.
Answer: [[right]]
Wrong since the query asks for position of person B with respect to A not the other way around

Sample response:

After considering all the previous movements, Person B's position is as follows:
Person B is at position 0 :
- initially at 1,
- then swapped with Person A,
- then swapped with Person C,
- then swapped with Person A again,
- and finally swapped with Person B
(Wrong: B can not swap with B!)
Person C is at position 1 :
- initially at 2,
- then swapped with Person A,
- then swapped with Person B,
- then swapped with Person A,
- then swapped with Person B,
- and finally swapped with Person C (Wrong again: C can not swap with C)
Since 0 is less than 1, Person B is to the left of Person C. Answer: [[left]]"

Swap. Sample input:

A starts at 0.
B starts at 1.
C starts at 2.
A swaps with C.
A swaps with B.
B swaps with A.
C swaps with B.
C swaps with A.
A swaps with C.
A swaps with C.
A swaps with C.
B swaps with A.
A swaps with B.
Where does C stand with respect to A?

Sample response:

B.2 Llama3 70B

Integer. Sample input:

A starts at 1.
B starts at 2.
C starts at 3.
B moves 2 to left.
A moves 7 to right.
C moves 8 to right.
A moves 4 to right.
C moves 3 to right.
A moves 5 to right.
C moves 16 to left.
B moves 5 to left.
C moves 20 to right.
A moves 2 to right.
Where does B stand with respect to A?

1. Person A starts at position 0, Person B starts at position 1, and Person C starts at position 2.
2. Person A swaps position with Person C, so Person A is now at position 2, and Person C is at position 0.
3. Person A swaps position with Person B, so Person A is now at position 1, and Person B is at position 2.
4. Person B swaps position with Person A, so Person B is now at position 1, and Person A is at position 2.
5. Person C swaps position with Person B, so Person C is now at position 1, and Person B is at position 0.
6. Person C swaps position with Person A, so Person C is now at position 2, and Person A is at position 1.

(the following is wrong!)

7. Person A swaps position with Person C (three times), but this doesn't change their positions, so Person A remains at position 1, and Person C remains at position 2.
8. Person B swaps position with Person A, so Person B is now at position 1, and Person A is at position 0.
9. Person A swaps position with Person B, so Person A is now at position 1, and Person B

Sample response:

Person A:
- starts at position
- moves 7 units to right: $1 + 7 = 8$
- moves 4 units to right: $8 + 4 = 12$
- moves 5 units to right: $12 + 5 = 17$
- moves 2 units to right: $17 + 2 = 19$
Person B:

is at position 0.
After all the movements, Person A is at position 1, and Person C is at position 2. Since 1 is smaller than 2, Person C is to the right of Person A. Answer: [[right]]"

B.3 Mixtral

Integer. Sample input:

A starts at 1.
B starts at 2.
C starts at 3.
B moves 6 to right.
C moves 4 to left.
A moves 6 to left.
C moves 6 to left.
A moves 16 to right.
B moves 4 to right.
A moves 20 to left.
C moves 4 to left.
A moves 26 to right.
B moves 25 to left.
Where does C stand with respect to A?

Sample response:

To answer this question, we first need to determine the current positions of Persons A and C after the given sequence of movements.
Person A's movements:
* Starts at position 1
* Moves 6 units to the left, making the new position $1 - 6 = -5$
(the following is wrong!)
* Moves 6 units to the left, making the new position $-5 - 6 = -11$
* Moves 16 units to the right, making the new position $-11 + 16 = 5$
* Moves 20 units to the left, making the new position $5 - 20 = -15$
* Moves 26 units to the right, making the new position $-15 + 26 = 11$
Person C's movements:
* Starts at position 3
* Moves 4 units to the left, making the new position $3 - 4 = -1$
* Moves 6 units to the left, making the new position $-1 - 6 = -7$
* Moves 4 units to the left, making the new position $-7 - 4 = -11$
* Moves 6 units to the left, making the new position $-11 - 6 = -17$
Now that we have the final positions of Persons A and C, we can compare them:
* Person A is at position 11
* Person C is at position -17
Since 11 is greater than -17, Person A is to the right of Person C.
The error in math did not matter, model is answering the wrong query regardless.
Therefore, the answer is: Answer: [[right]]"

Swap. Sample input:

A starts at 0.
B starts at 1.
C starts at 2.
A swaps with C.
B swaps with C.
B swaps with C.
B swaps with C.
A swaps with C.
C swaps with B.
B swaps with A.
B swaps with A.
C swaps with B.
A swaps with C.
Where does A stand with respect to C?

Sample response:

The model is making so many incorrect statements since it is trying to aggregate all swap statements for each entity.
Person A starts at position 0, after all the specified movements, A swaps positions with C twice, and *(wrong!)*
B swaps positions with A three times. *(wrong!)*
This means A and B essentially swap positions. *(wrong!)*
So, since B ends up at position 0 (by swapping with A), *(correct but by chance!)*
and C is at position 2, *(wrong!)*
then A must be at position 1.
» logical conclusion based on wrong premises
Therefore, with respect to Person C, who is at position 2, Person A is to the left. Answer: [[left]]"

C Impact of Query Type.

According to the results illustrated in Tables 4 and 3, the query type has a greater impact on tasks involving five individuals compared to those with three. This may be attributed to the fact that a single update statement is more likely to alter the relative positions of all entities, thus inherently achieving state dependence without the need to enforce it.

Figure 5 compares system performance on "state-dependent" versus "random" queries for tasks involving five entities. GPT-4o demonstrates near-perfect performance across both query types, emerging as the top-performing model. In contrast, the least effective model being the small variant of LLaMA3 barely outperforms a random baseline. For these two extremes, the difference in performance between query types is negligible. However, the performance gap between query types is most pronounced in models that excel at lower depths but struggle at higher depths, particularly the systems based on Mixtral model and the large variant of

LLaMA3. These models find the state-dependent setting more challenging, as expected.

D Detailed Result Tables

In this section, a complete table of the results of our controlled experiments is presented.

System	#	Query	Update	Depth										
				0	1	2	3	4	5	6	7	8	9	10
GPT 3.5	3	Ran	Swap	0.96	0.54	0.48	0.40	0.42	0.44	0.50	0.40	0.54	0.54	0.52
	3	Dep	Swap	0.74	0.56	0.50	0.48	0.40	0.42	0.56	0.58	0.50	0.44	0.48
	3	Dep	Int	0.86	0.80	0.86	0.90	0.78	0.72	0.66	0.68	0.72	0.58	0.67
	5	Ran	Swap	0.88	0.54	0.54	0.48	0.42	0.50	0.52	0.46	0.50	0.48	0.52
	5	Dep	Swap	0.80	0.60	0.58	0.42	0.44	0.50	0.50	0.40	0.42	0.56	0.42
	5	Dep	Int	0.84	0.78	0.64	0.70	0.60	0.56	0.62	0.66	0.52	0.66	0.60
	5	Dep	L-Int	0.82	0.62	0.72	0.58	0.64	0.60	0.68	0.62	0.56	0.54	0.64
GPT 3.5 CoT	3	Ran	Swap	0.98	0.86	0.56	0.60	0.48	0.44	0.48	0.38	0.56	0.44	0.48
	3	Dep	Swap	0.94	0.78	0.60	0.56	0.54	0.46	0.50	0.44	0.60	0.42	0.42
	3	Dep	Int	0.92	0.96	0.90	0.90	0.84	0.86	0.80	0.80	0.64	0.64	0.70
	5	Ran	Swap	0.90	0.88	0.65	0.59	0.54	0.50	0.52	0.54	0.44	0.50	0.50
	5	Dep	Swap	0.94	0.82	0.62	0.64	0.50	0.50	0.54	0.48	0.46	0.52	0.50
	5	Dep	Int	0.98	0.98	0.80	0.86	0.88	0.72	0.76	0.60	0.58	0.62	0.66
	5	Dep	L-Int	0.98	0.90	0.88	0.82	0.86	0.80	0.78	0.78	0.72	0.70	0.72
GPT 4.0	3	Ran	Swap	0.90	1.00	1.00	1.00	0.96	1.00	1.00	1.00	1.00	0.98	1.00
	3	Dep	Swap	1.00	0.92	0.98	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00
	3	Dep	Int	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	5	Ran	Swap	1.00	0.98	1.00	0.96	1.00	1.00	1.00	1.00	0.98	1.00	0.94
	5	Dep	Swap	1.00	0.98	1.00	1.00	1.00	1.00	0.96	1.00	1.00	1.00	0.96
	5	Dep	Int	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	0.96
	5	Dep	L-Int	0.98	1.00	1.00	0.96	1.00	0.98	1.00	1.00	0.98	1.00	1.00
GPT 4.0 CoT	3	Ran	Swap	1.00	0.98	1.00	1.00	0.96	1.00	1.00	0.94	1.00	0.96	0.98
	3	Dep	Swap	1.00	1.00	0.98	1.00	1.00	0.96	0.96	1.00	0.94	1.00	1.00
	3	Dep	Int	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	0.96	1.00
	5	Ran	Swap	0.98	1.00	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	5	Dep	Swap	1.00	1.00	0.98	1.00	0.96	1.00	1.00	1.00	0.98	1.00	0.96
	5	Dep	Int	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98
	5	Dep	L-Int	0.98	1.00	1.00	0.94	1.00	1.00	0.98	1.00	1.00	0.96	1.00

Table 3: Accuracy at various depths of GPT models on LinearWorld task variations. Models evaluated with Chain-of-Thought prompting are denoted by "CoT" in their names. For the query types, "Ran" refers to "random" and "Dep" to "dependent" query type. Also, regarding the update types, "Int" and "L-Int" refer to "integer" and "large integer" displacement update types, respectively.

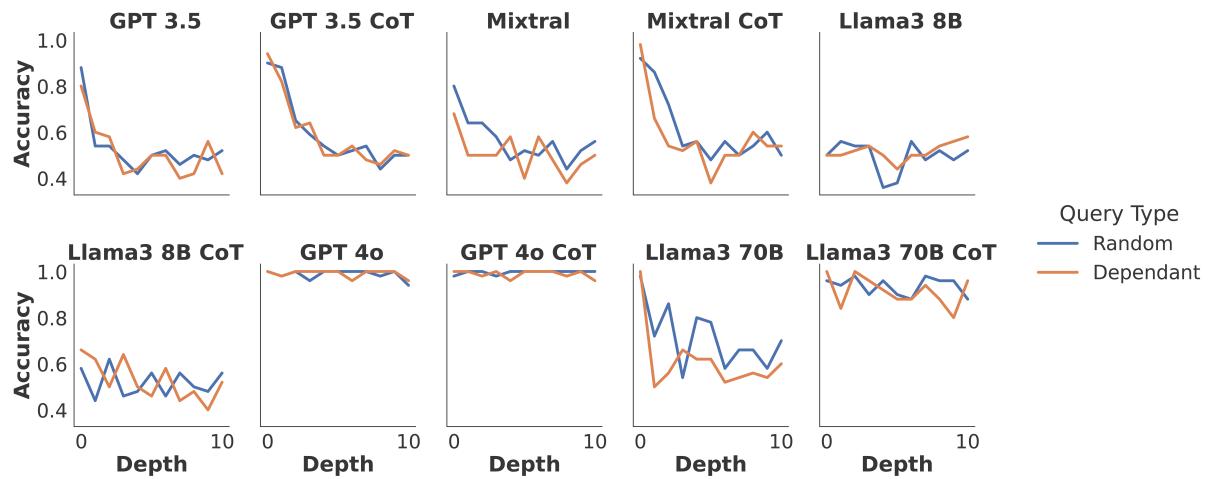


Figure 5: Accuracy at different depths comparing the "state-dependant" query type with the "random" query type in the LinearWorld task (the variants with 5 individuals) accross all the systems.

System	#	Query	Update	Depth										
				0	1	2	3	4	5	6	7	8	9	10
Mixtral	3	Ran	Swap	0.80	0.68	0.48	0.48	0.58	0.44	0.46	0.42	0.56	0.44	0.40
	3	Dep	Swap	0.58	0.58	0.54	0.50	0.50	0.42	0.46	0.48	0.46	0.58	0.60
	3	Dep	Int	0.44	0.78	0.68	0.64	0.54	0.46	0.44	0.44	0.38	0.24	0.28
	5	Ran	Swap	0.80	0.64	0.64	0.58	0.48	0.52	0.50	0.56	0.44	0.52	0.56
	5	Dep	Swap	0.68	0.50	0.50	0.50	0.58	0.40	0.58	0.48	0.38	0.46	0.50
	5	Dep	Int	0.62	0.74	0.74	0.42	0.54	0.40	0.44	0.38	0.38	0.46	0.32
	5	Dep	L-Int	0.72	0.74	0.66	0.44	0.64	0.50	0.42	0.50	0.66	0.60	0.56
Mixtral CoT	3	Ran	Swap	0.90	0.82	0.54	0.54	0.48	0.58	0.42	0.40	0.54	0.48	0.42
	3	Dep	Swap	1.00	0.74	0.60	0.68	0.52	0.54	0.60	0.36	0.58	0.42	0.62
	3	Dep	Int	0.68	0.78	0.92	0.82	0.86	0.86	0.82	0.78	0.84	0.82	0.78
	5	Ran	Swap	0.92	0.86	0.72	0.54	0.56	0.48	0.56	0.50	0.54	0.60	0.50
	5	Dep	Swap	0.98	0.66	0.54	0.52	0.56	0.38	0.50	0.50	0.60	0.54	0.54
	5	Dep	Int	0.80	0.82	0.82	0.88	0.88	0.88	0.82	0.72	0.72	0.82	0.76
	5	Dep	L-Int	0.90	0.80	0.70	0.78	0.80	0.78	0.68	0.60	0.68	0.72	0.80
Llama3 8B	3	Ran	Swap	0.46	0.50	0.50	0.42	0.50	0.54	0.48	0.46	0.52	0.56	0.44
	3	Dep	Swap	0.50	0.56	0.50	0.54	0.48	0.52	0.46	0.46	0.42	0.44	0.52
	3	Dep	Int	0.60	0.54	0.68	0.80	0.72	0.74	0.70	0.64	0.72	0.82	0.62
	5	Ran	Swap	0.50	0.56	0.54	0.54	0.36	0.38	0.56	0.48	0.52	0.48	0.52
	5	Dep	Swap	0.50	0.50	0.52	0.54	0.50	0.44	0.50	0.50	0.54	0.56	0.58
	5	Dep	Int	0.38	0.78	0.54	0.72	0.52	0.68	0.60	0.62	0.56	0.54	0.60
	5	Dep	L-Int	0.68	0.58	0.52	0.50	0.62	0.50	0.58	0.56	0.52	0.54	0.54
Llama3 8B CoT	3	Ran	Swap	0.70	0.52	0.58	0.44	0.52	0.56	0.54	0.64	0.52	0.42	0.52
	3	Dep	Swap	0.56	0.50	0.62	0.52	0.54	0.38	0.58	0.44	0.56	0.32	0.48
	3	Dep	Int	0.36	0.82	0.80	0.76	0.66	0.64	0.70	0.70	0.74	0.70	0.62
	5	Ran	Swap	0.58	0.44	0.62	0.46	0.48	0.56	0.46	0.56	0.50	0.48	0.56
	5	Dep	Swap	0.66	0.62	0.50	0.64	0.50	0.46	0.58	0.44	0.48	0.40	0.52
	5	Dep	Int	0.46	0.74	0.78	0.66	0.58	0.70	0.52	0.58	0.60	0.58	0.48
	5	Dep	L-Int	0.46	0.68	0.56	0.62	0.62	0.60	0.54	0.52	0.46	0.46	0.42
Llama3 70B	3	Ran	Swap	1	0.82	0.54	0.34	0.56	0.66	0.52	0.52	0.62	0.56	0.52
	3	Dep	Swap	1	0.48	0.48	0.5	0.58	0.62	0.42	0.54	0.44	0.5	0.56
	3	Dep	Int	0.94	1	0.96	0.96	0.94	0.96	0.96	0.98	0.9	0.92	0.88
	5	Ran	Swap	0.98	0.72	0.86	0.54	0.8	0.78	0.58	0.66	0.66	0.58	0.7
	5	Dep	Swap	1	0.5	0.56	0.66	0.62	0.62	0.52	0.54	0.56	0.54	0.6
	5	Dep	Int	0.98	0.94	0.9	0.94	0.96	0.98	0.92	0.96	0.96	0.94	0.98
	5	Dep	L-Int	0.9	0.94	0.98	0.96	0.9	0.9	0.92	0.96	0.94	0.88	0.82
Llama3 70B CoT	3	Ran	Swap	0.96	0.86	0.88	0.8	0.78	0.9	0.82	0.86	0.76	0.72	0.84
	3	Dep	Swap	1	0.76	0.9	0.94	0.9	0.88	0.94	0.82	0.76	0.78	0.84
	3	Dep	Int	0.94	1	1	1	0.96	0.96	0.9	0.96	0.94	0.96	0.92
	5	Ran	Swap	0.96	0.94	0.98	0.9	0.96	0.9	0.88	0.98	0.96	0.96	0.88
	5	Dep	Swap	1	0.84	1	0.96	0.92	0.88	0.88	0.94	0.88	0.8	0.96
	5	Dep	Int	0.98	1	1	0.98	1	1	1	0.98	0.96	0.98	0.94
	5	Dep	L-Int	0.96	0.92	0.9	0.92	0.94	0.96	0.9	0.94	0.88	0.84	0.92

Table 4: Accuracy at various depths of open-source models on LinearWorld task variations. Models evaluated with Chain-of-Thought prompting are denoted by "CoT" in their names. For the query types, "Ran" refers to "random" and "Dep" to "dependent" query type. Also, regarding the update types, "Int" and "L-Int" refer to "integer" and "large integer" displacement update types, respectively.