LTLBench: Towards Benchmarks for Evaluating Temporal Reasoning in Large Language Models

Anonymous ACL submission

Abstract

Temporal Reasoning (TR) is a critical ability for LLMs to understand and reason over temporal information and relationships between events. To study the TR ability in LLMs, prior works provide different ways for evaluating various aspects of TR ability. In this work, we propose an alternative perspective for evaluating TR ability by leveraging Linear Temporal Logic (LTL), and develop a pipeline to automatically synthesize challenges for assessing the TR ability of LLMs. Based on this pipeline, we construct a dataset, namely LTLBench, consisting of 2000 TR challenges, and benchmark 12 LLMs across 5 different methods. Furthermore, we conduct additional experiments to investigate the impact of increasing the number of formula operators and events on both LLM performance and the complexity of TR problems. We also perform qualitative analyses of their reasoning processes and the effects of varying the number of events and formula operators, which reveal 3 main issues in their temporal reasoning processes and the unexpected performance changes observed as problem complexity increases. We expect this work to provide valuable insights into the TR ability of LLMs[1].

1 Introduction

Temporal Reasoning (TR) is a critical reasoning ability of LLMs, encompassing the understanding, processing, and reasoning over temporal information and relationships between events, which is essential for solving problems across diverse scenarios (Shoham and Goyal, 1988; Chittaro and Montanari, 2000; Vila, 1994). Prior studies have demonstrated that although LLMs show some promise in TR, they still struggle with TR and a substantial performance gap persists between the state-of-the-art LLMs and humans on TR (Chu et al., 2023; Wang and Zhao, 2024; Beniwal et al., 2024). Nevertheless, since TR encompasses various aspects, existing investigations of the TR ability of LLMs remain incomplete. Therefore, in this work, we adopt Linear Temporal Logic to explore TR from a new perspective, focusing on formal logical reasoning over temporal information.

Linear Temporal Logic (LTL) is a formal logic, specifically a modal temporal logic, that is widely used and studied for expressing and reasoning about sequences of events over time (Kröger and Merz, 2008; Goranko and Rumberg, 2024). Although complex LTL is typically discussed and used in the context of formal and program verification, basic and moderately complex LTL tasks are, in fact, ubiquitous in daily tasks. For example, if people are out of milk, they will eventually buy it, which can be formalized in an LTL formula as $G(\text{OutOfMilk} \rightarrow F(\text{BuyMilk}))$. Likewise, if the traffic light is green, it will then turn to yellow, which can be formalized as $G(\text{Green} \rightarrow X(\text{Yellow}))$[2]. From this perspective, LTL provides a natural and formal way to represent and operate on temporal relations between events. Therefore, we leverage LTL as a main component to construct TR problems.

To explore the TR ability of LLMs, we propose a novel approach, a TR challenges generation pipeline, to automatically synthesize TR problems. Each generated TR problem mainly consists of a context that depicts the situation of a TR problem and a hypothesis that requires LLMs to judge its validity against the given context. The core components of the pipeline involve a randomly generated directed graph that serves as the preparation for subsequent problem generations, a randomly generated LTL formula that acts as the hypothesis for the given context, and the NuSMV model

[1] Our code is open-sourced and available at https://anonymous.4open.science/r/LTLBench-Anonymous/.

[2] Refer to Section 2 for more details on LTL syntax and semantics.

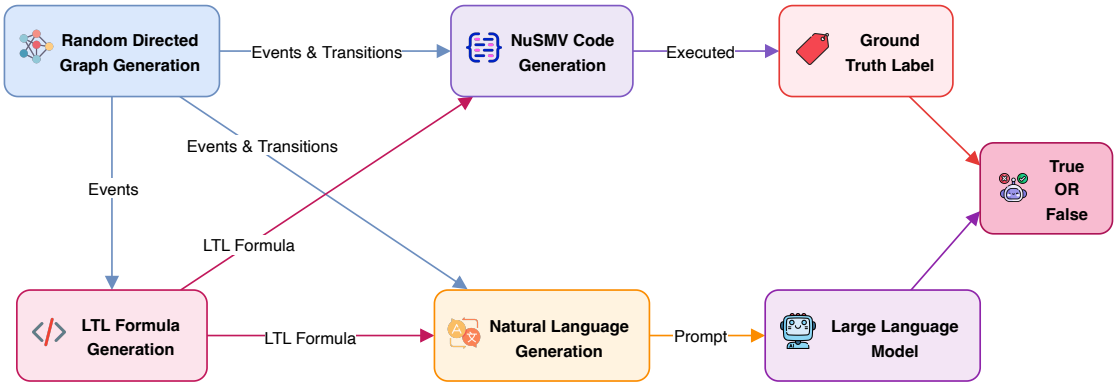


Figure 1: The overview of the TR problem generation pipeline.

checker (Cimatti et al., 2002) which executes the code of events transitions and the LTL formula to provide the ground truth label. As shown in Figure 1, during the generation process for a TR problem, we first generate a random directed graph. Then, we adopt and slightly modify the LTL formulas generation algorithm designed by Zhu (2021) to generate an LTL formula based on the events given in the graph. Subsequently, both the event information and the LTL formula are converted into NuSMV (Cimatti et al., 2002) code and executed to obtain the ground truth label. Finally, the event information and the LTL formula are translated into a TR problem in the form of natural language.

Furthermore, to conduct an intensive and comprehensive evaluation, we generate a total of 2000 TR challenges using our proposed pipeline, referred to as LTLBench, and we evaluate 12 LLMs across 5 different methods. We not only demonstrate their TR ability but also provide several qualitative insights that reveal 3 main issues they fail to reason over these challenges. In addition, we conduct additional experiments to investigate the impact of varying the number of events and operators on both the performance of LLMs and the complexity of TR problems. The key contributions of our study are summarized as follows:

1. We develop a novel TR challenges generation pipeline, which lays on Linear Temporal Logic, to evaluate the TR ability of LLMs from a new perspective, focusing on formal logical reasoning over temporal information;

2. Based on the pipeline, we construct a dataset, LTLBench, consisting of 2000 TR challenges, and evaluate 12 LLMs across 5 different methods, providing both quantitative results and qualitative insights which reveal 3 main issues of their reasoning failures;

3. We further conduct two additional experiments to demonstrate that increasing the number of formula operators and events leads to more challenges for LLMs, and offer qualitative insights to discuss the unexpected performance changes observed as problem complexity increases.

2 Preliminary

Since we leverage LTL, which is a formal language for reasoning about linear-time propositions and their temporal relationships, to formulate hypotheses in temporal reasoning problems in this work, we provide a concise overview of LTL syntax and semantics to establish the necessary background as follows.

The syntax of LTL is defined over a set of atomic propositions $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, logical operators, and temporal operators. Formally, LTL formulas are constructed according to the following grammar in Backus–Naur form[3]:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid X\phi \mid F\phi \mid G\phi$$

where p is an atomic proposition; $\neg$, $\wedge$, $\vee$, and $\rightarrow$ are the standard logical operators for negation, conjunction, disjunction, and implication, respectively; $X\phi$ indicates that ϕ holds at the next time step; $F\phi$ indicates that ϕ eventually holds at some future time step; $G\phi$ indicates that ϕ holds at all future time steps.

[3]For a concise explanation, we ignore several operators such as U, R, W, and M that we do not use in this work.

The formal semantics of LTL are defined over Kripke structures (Kripke, 1963). A Kripke structure is a tuple:

$$M = (S, S_0, R, V),$$

where S is a finite set of states; $S_0 \subseteq S$ is the set of initial states; $R \subseteq S \times S$ is a transition relation, in which $\forall s \in S$, $\exists s' \in S$ such that $(s, s') \in R$; $V : S \to 2^{\mathcal{P}}$ is a valuation function that specifies, for each state, which atomic propositions in $\mathcal{P}$ are true in s.

A path $\pi = [s_0, s_1, s_2, \ldots]$ in M is an infinite sequence of states such that $s_0 \in S_0$ and $(s_i, s_{i+1}) \in R$ for all $i \geq 0$. Let $\pi_i = [s_i, s_{i+1}, \ldots]$ denote the suffix of π starting at position i.

The satisfaction relation $M, \pi \models \phi$ is defined inductively as follows:

- $M, \pi \models p$ iff $p \in V(s_0)$;
- $M, \pi \models \neg\phi$ iff $M, \pi \not\models \phi$;
- $M, \pi \models \phi_1 \wedge \phi_2$ iff $M, \pi \models \phi_1$ and $M, \pi \models \phi_2$;
- $M, \pi \models \mathsf{X}\phi$ iff $M, \pi_1 \models \phi$;
- $M, \pi \models \mathsf{F}\phi$ iff $\exists k \geq 0$ such that $M, \pi_k \models \phi$;
- $M, \pi \models \mathsf{G}\phi$ iff $\forall k \geq 0,\ M, \pi_k \models \phi$.

We say $M \models \phi$ if for all paths π starting from initial states in M, $M, \pi \models \phi$.

3 Related Work

3.1 TR in LLMs

Temporal Reasoning in LLMs has recently obtained substantial attention (Xiong et al., 2024; Fatemi et al., 2024; Beniwal et al., 2024; Chu et al., 2023; Hu et al., 2023; Liu et al., 2023; Vashishtha et al., 2020). Beniwal et al. (2024) point out notable deficiencies of LLMs in understanding and reasoning over temporal information and reasoning, while Xiong et al. (2024) propose TG-LLM, a framework aimed at improving LLMs performance on TR tasks. Furthermore, Liu et al. (2025) introduce a Time-R1, which employs reinforcement learning fine-tuning to enhance temporal reasoning ability of LLMs, enabling smaller models to match or even surpass larger ones. While these works collectively highlight the progress and importance of TR ability in LLMs, the exploration of TR ability in LLMs remains ongoing, requiring further efforts to reveal and understand the boundaries.

3.2 TR Benchmarks

To discover and evaluate the TR ability of LLMs, a variety of TR datasets and benchmarks have been proposed, targeting different aspects of TR at varying levels of complexity (Fatemi et al., 2024; Wang and Zhao, 2024; Xiong et al., 2024; Beniwal et al., 2024; Qin et al., 2021; Tan et al., 2023; Virgo et al.). For example, Xiong et al. (2024) construct a TR dataset by leveraging a large temporal knowledge graph, YAGO11k (Dasgupta et al., 2018), and utilizing GPT-3.5 and rule-based Python scripts to generate TR challenges. Additionally, Fatemi et al. (2024) employ random graph generation as a foundation and preparation to form rule-based and different types of temporal facts and questions without introducing LLMs to generate TR tasks, focusing on temporal semantics and arithmetic reasoning and proposing a benchmark called Test of Time. Furthermore, Wang and Zhao (2024) introduce a TR dataset consisting of various temporal aspects such as order, arithmetic, frequency, and duration. While these approaches provide valuable perspectives for evaluating TR ability, they generally lack systematic support for representing and reasoning with temporal operators such as *eventually* and *always*, as well as their complex compositions. By contrast, our work that leverages LTL enables a natural support over those temporal operators and can explore TR ability from another perspective.

4 TR Problem Generation Pipeline

The pipeline to generate a single TR problem consists of four stages: (1) Random Directed Graph Generation, (2) LTL Formula Generation, (3) NuSMV Code Generation, and (4) Natural Language Generation. We demonstrate the overview of the process for a TR problem generation in Figure 1.

4.1 Random Directed Graph Generation

During this stage, a directed graph is randomly generated with a given number of events n where $n > 1$ to ensure the formation of transitions between events.

In this graph, each $node_i$ represents an individual $event_i$, and each $edge_j^i$ is a directed edge pointing from $node_i$ to another $node_j$, which forms the relationships and transitions between events, indicating that $event_j$ represented by $node_j$ occurs after the $event_i$ represented by $node_i$. It is important to note that each $node_i$ within the graph can

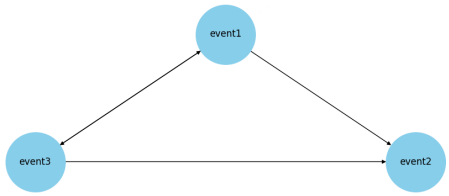


Figure 2: An example of a generated random directed graph.

have multiple outgoing edges, signifying that several subsequent events can follow $event_i$, as well as multiple incoming edges, indicating that $event_i$ can be preceded by several other events. To form a clear semantics aligned with the follow-up stages, for an event $event_i$ followed by only one event $event_j$, we say that after $event_i$, $event_j$ must happen, while for an event $event_i$ followed by more than one event $event_1, event_2, \ldots$, we say that after $event_i$, $event_1, event_2, \ldots$ can happen.

As an example illustrated in Figure 2, given $n = 3$, three events are generated: $event_1$, $event_2$, and $event_3$. The case that $event_1$ points to $event_2$ indicates that $event_2$ can happen after $event_1$. The case that $event_1$ points to $event_3$ and $event_3$ also points to $event_1$ means that $event_1$ can happen after $event_3$ and $event_3$ can happen after $event_1$. In addition, to note, $event_1$ not only points to $event_2$ but also $event_3$, indicating that both $event_2$ and $event_3$ can happen after $event_1$. By contrast, if $event_1$ only points to $event_2$, it indicates that $event_2$ must happen after $event_1$.

The generation of the random directed graph during this stage serves as the foundation and preparation for generating the LTL formula, NuSMV code, and TR problem represented in natural language by providing the information of events and also the transitions between events.

4.2 LTL Formula Generation

Based on the events generated in the graph, we employ and slightly modify the algorithm designed by Zhu (2021) to generate an LTL formula with a given number of operators m where $m > 0$. The LTL operators include unary and binary operators. Unary operators, for example, include but are not limited to X which indicates that for a given event ϕ, $X\phi$ denotes that the event ϕ will occur at the next moment, and F for which $F\phi$ means that event ϕ will eventually occur at some point in the future. Binary operators include but are not limited to & representing logical AND and | representing logical OR. The given number of operators refers to the number of unary and binary LTL operators contained in an LTL formula. The algorithm for generating LTL formulas is detailed in Algorithm 1 in Appendix A.

```
(event1 -> (G (F event2)))
```

Listing 1: An example of a generated LTL formula.

An example of a generated LTL formula is also provided and shown in Listing 1. The formula means that if $event_1$ happens, it will be globally true that, at some point in the future, $event_2$ will eventually happen.

The generation of the LTL formula aims to be the preparation for generating the LTLSPEC part of the NuSMV code and also to generate the hypothesis part of the TR problem represented in natural language.

4.3 NuSMV Code Generation

Given the information of events from the graph and the LTL formula, this stage converts them into NuSMV code, which consists of two parts: (1) context generation and (2) LTLSPEC generation. The context describes the situation of the TR problem, while LTLSPEC represents a hypothesis regarding the context.

For context generation, it includes event definitions, initial event setup, and event transitions setup. Based on the generated graph, events and their transitions are converted into the context part of the NuSMV code, and an initial event is randomly selected. As shown in Listing 2, Lines 2-3 define three events, Line 5 specifies the initial event which is $event_3$, and Lines 6-11 construct the event transitions, in which, for example, Line 8 indicates that $event_2$ and $event_3$ can follow $event_1$, while $event_2$ remains to itself and after $event_2$, no other events can happen.

For LTLSPEC generation, the generated LTL formula is translated into the NuSMV code, as illustrated at Line 13 in Listing 2 which represents the LTL formula shown in Listing 1.

The resulting NuSMV code is executed by the NuSMV model cheker during the generation process to obtain the ground truth label for the TR problem.

```
MODULE main
VAR
    state : {event1, event2, event3};
ASSIGN
    init(state) := event3;
    next(state) := case
        state = event1 : {event2, event3};
        state = event2 : event2;
        state = event3 : {event1, event2};
    esac;
LTLSPEC ((state=event1) -> (G (F (state=event2))))
```

Listing 2: An example of NuSMV code.

```
=== Context ===

Initially, event3 happened. After event1, event2 and event3 can happen. After event2, no other events can happen. After event3, event1 and event2 can happen.

=== Hypothesis ===

C1: Event2 eventually happens.
C2: C1 always holds.
C3: If event1 happens, then C2 holds.

C3 is True or False?
```

Listing 3: An example of TR problem in the form of natural language.

4.4 Natural Language Generation

During this stage, the events information in the graph and the LTL formula are converted into natural language. Similar to the NuSMV code generation, this stage consists of two parts: (1) context generation and (2) hypothesis generation. The context describes the problem situation while the hypothesis specifies what LLMs are required to determine given the context.

For context generation, based on the generated graph and the initial event, event information and transitions are converted into natural language. As shown at Lines 1-4 in Listing 3, the initial event and events transitions are represented in natural language.

For hypothesis generation, the generated LTL formula is transformed into natural language as shown at Lines 5-10 in Listing 3. Additionally, Line 11 is used to prompt LLMs to determine the validity of the hypothesis.

The TR problem represented in natural language generated in this stage is the core and final product of the generation process, which will be used to evaluate the TR ability of LLMs.

5 Experiment Settings

To evaluate the TR ability of LLMs, based on the pipeline, we construct a dataset, LTLBench, consisting of 2000 problems. Each problem is generated with a fixed number of events $n = 3$ and formula operators $m = 3$. Additionally, to explore the impact of changes in the number of formula operators, we conduct evaluations on newly generated problems with a fixed number of events $n = 2$ while varying the number of formula operators $m \in \{1, 2, 3, 4, 5, 7, 9\}$. For each (n, m_i), such as $(2, 1)$ indicating that the number of events is 2 and the number of operators is 1, we generate 300 problems as a dataset for evaluation. Similarly, to examine the effects of the number of events, we fix the number of formula operators $m = 2$ and vary the number of events $n \in \{2, 3, 4, 5, 7, 9\}$. For all generated datasets, their distributions of ground truth labels are meticulously balanced, with half of the problems labeled as *True* and the other half of the problems as *False*. Furthermore, we mainly adopted Accuracy, defined as the proportion of the correctly answered problems, as the primary evaluation metric.

For comprehensive evaluations, we select a total of 12 LLMs. Specifically, we evaluate DeepSeek-V3 (DeepSeek-AI et al., 2025) in both non-thinking mode and thinking mode, which are officially noted as *DeepSeek-Chat* model and *DeepSeek-Reasoner* model. We also include 3 OpenAI models, which are *GPT-3.5-Turbo* (Brown et al., 2020), *GPT-4o-Mini* (OpenAI et al., 2024), and *GPT-5-Mini*. In addition, we select 4 Qwen models (Yang et al., 2025; Qwen et al., 2025), namely *Qwen-Turbo*[4], *Qwen3:32B*, *Qwen3:14B*, and *Qwen2.5:72B-Instruct*. Furthermore, we also include 3 additional models, which are *Gemma3:12B-Instruct* (Team et al., 2025), *Mistral:7B-Instruct* (Jiang et al., 2023), and *Phi4:14B* (Abdin et al., 2024).

Moreover, to examine how different methods affect LLMs performance on TR challenges, we adopt 5 methods, which are *Direct Prompting*, *Zero-Shot CoT* (Kojima et al., 2022), *Few-Shot CoT* (Wei et al., 2022), *Self-Consistency* (Wang et al., 2023), and *Least-to-Most* (Zhou et al., 2022).

[4]The exact version of *Qwen-Turbo* is *Qwen-Turbo-2025-04-28*.

Direct Prompting directly queries an LLM to generate answer in *True* or *False* for a given TR problem. *Zero-Shot CoT* prompts an LLM to reason step by step without examples, whereas *Few-Shot CoT* provides 2 examples with each consisting of a TR problem, thinking process, and final answer. *Self-Consistency* extends *Zero-Shot CoT* by sampling 3 reasoning paths and taking majority voting to select the final answer. In *Least-to-Most*, an LLM is first prompted to answer a set of breakdown smaller questions and is then prompted again with the response to previous questions to reason on the main problem to give the final answer. For *Direct Prompting*, *Zero-Shot CoT*, *Few-Shot CoT*, and *Least-to-Most* methods, we set the *temperature* to 0 and the *max completion tokens* to 2000 for all models. For *Self-Consistency*, since it requires diversity of candidates, we set the *temperature* to 0.7 and the *max completion tokens* to 2000 for all models[5]. The prompt templates for *Direct Prompting*, *Zero-Shot CoT*, and *Few-Shot CoT* are provided in Prompt Template 1, 2, and 3 in Appendix A, respectively, with *Self-Consistency* using the same prompt template as *Zero-Shot CoT*. The prompt templates for the two stages of *Least-to-Most* are shown in Prompt Template 4 and 5 in Appendix A.

6 Results and Analyses

6.1 Evaluation with LTLBench

We evaluate the selected 12 LLMs on LTLBench consisting of 2000 generated TR problems with the number of events $n = 3$ and formula operators $m = 3$. The results of all models equipped with different methods are reported in Table 1. We also demonstrate the performance of each model with different methods in a grouped horizontal bar chart as shown in Figure 3 in Appendix A for a better visual comparison of the results.

From the results, among all models and methods, the best one is the *GPT-5-Mini* with the *Few-Shot CoT* method, achieving an accuracy of 93.95%, whereas the worst one is the *GPT-3.5-Turbo* with the *Least-to-Most* method, whose accuracy of 51.05% is only slightly above random guessing. In addition, the average accuracies of each method across models are 65.83% for *Direct Prompting*, 67.18% for *Zero-Shot CoT*, 76.33% for *Few-Shot CoT*, 68.29% for *Self-Consistency*, and 68.21% for *Least-to-Most*, in which *Few-Shot CoT* demonstrates its significant performance improvement while the other three methods slightly outperform the *Direct Prompting*. As demonstrated in both Table 1 and Figure 3, *Few-Shot CoT* consistently and significantly outperforms other methods regardless of models. In addition, *Self-Consistency* tends to fail to further improve the performance of recent high-performing models regardless of the parameter sizes, such as *DeepSeek-Reasoner*, *DeepSeek-Chat*, *GPT-5-Mini*, *GPT-4o-Mini*, *Qwen-Turbo*, *Qwen3:32B*, *Qwen3:14B*, and *Phi4:14B*, while for earlier models such as *Mistral:7B-Instruct*, *GPT-3.5-Turbo*, and *Qwen2.5:72B-Instruct*, it can still significantly enhance their performance. We consider the reason is that after multiple iterations of LLMs, recent LLMs approach a problem in a more self-consistent way, so that the *Self-Consistency* method may fail to further improve the performance on these models. In addition, across most models, *Zero-Shot CoT* can only marginally increase the performance, which could be attributed to that most of the evaluated models have already incorporated reasoning or thinking instructions during their pretraining or instruction-tuning process, such as DeepSeek series models, OpenAI's GPT-5 series models, and also Qwen3 series models, so that even in the case of using *Direct Prompting*, they automatically invoke the chain-of-thought to reason over the problems, as our observation of their responses.

In addition, notably, we observe that *DeepSeek-Chat* outperforms *DeepSeek-Reasoner*. Typically, we believe models equipped with enhanced and polished reasoning ability should outperform ones without it. From our perspective, this unexpected result may arise because our LTL-based TR challenges require a more complex formal logic reasoning along with temporal reasoning, in which the reason steps may be different from other challenges without formal logic injected in such as Test of Time (Fatemi et al., 2024) and may cause untypical reasoning steps. The reasoning steps that *DeepSeek-Reasoner* are trained on during pretraining and perform during inference may not align with the reasoning steps demanded by LTL-based TR problems. Thereby, although it is explicitly asked to think in a specific way to tackle LTL-based TR problems, the pretrained distribution of output tokens of reasoning steps is not suitable for this kind of problem and could even further disturb the reasoning steps to some extent, leading to even worse than equipping non-thinking or non-

[5]For the GPT-5 series models, the *temperature* is not settable, so we do not use *temperature* for *GPT-5-Mini*.

Model	DP	ZS CoT	FS CoT	SC	L2M
DeepSeek-Reasoner	71.55	69.95	81.10	73.70	72.75
Deepseek-Chat	**80.10**	**78.65**	90.40	79.40	**81.05**
GPT-5-Mini	79.70	78.45	**93.95**	**80.53**	77.75
GPT-4o-Mini	63.50	62.60	71.55	63.25	63.70
GPT-3.5-Turbo	53.25	56.55	60.40	57.10	51.05
Qwen-Turbo	66.50	67.60	78.70	68.90	69.60
Qwen2.5:72B-Instruct	59.05	68.10	76.05	67.75	69.90
Qwen3:32B	66.45	66.65	76.50	68.10	70.45
Qwen3:14B	67.45	67.45	75.15	67.80	71.25
Phi4:14B	66.65	65.60	73.50	65.90	66.15
Gemma3:12B-Instruct	60.15	67.20	78.40	68.20	67.20
Mistral:7B-Instruct	55.60	57.35	60.25	59.00	57.70

Table 1: The accuracy (%) of LLMs equipped with different methods evaluated on LTLBench, in which DP stands for *Direct Prompting*, ZS CoT for *Zero-Shot CoT*, FS CoT for *Few-Shot CoT*, SC for *Self-Consistency*, and L2M for *Least-to-Most*, and the best accuracy across models for each method is highlighted in bold.

reasoning models with explicit task-specific CoT methods.

Moreover, to get more insights into the reasoning process of LLMs on the TR challenges, we further conduct a qualitative analysis. For each model paired with each method, we randomly select 10 problem and answer pairs, thus resulting in a total of 600 pairs. We then filter out all pairs with correct answers and finally retain 145 pairs with incorrect answers, to focus on analyzing the potential factors that lead LLMs to fail in answering problems correctly. Among these 145 pairs, we identify three major findings. The first is *Temporal Semantics and Reasoning Misalignment*, which indicates that although LLMs demonstrate an understanding of temporal semantics during reasoning steps, they may still apply these semantics incorrectly when performing actual reasoning. A representative example is observed in the results of *GPT-4o-Mini* with the *Few-Shot CoT* method, in which although it understands that after *event 1*, the *event 3* must happen, which is then followed by *event 2*, when it reasons about whether *event 2* happens at next state after *event 1*, it does not precisely capture that the exact next state is *event 3* but it reasons in a misaligned way such that since that *event 1* is followed by *event 3* and *event 3* is followed by *event 2*, *event 2* happens at next state of *event 1*, which is however incorrect[6]. The second is *Context Hypothesis Detachment*, which indicates that LLMs largely ignore the context and only leverage information from the hypothesis to perform reasoning. A representative example is observed from the results of *Qwen-Turbo* with the *Direct Prompting* method, in which since *C1* and *C2* are identical, when asked whether both *C1* and *C2* hold, it reasons in a shortcut way without leveraging context information about the transitions of events, and directly gives the answer that *C3* is true, which however is not. We observe this happens in several models such as *Qwen-Turbo*, *GPT-4o-Mini*, *Mistral:7B-Instruct*, etc[7]. The third is *Reasoning Error Amplification*, which means the previous reasoning errors without retrospection for correcting lead to the later reasoning ignoring the errors and using the previous incorrect steps or conclusions to future reason, resulting in that the reasoning errors are amplified during the reasoning process. This is a normal and typical error when LLMs perform reasoning and we observe it in the reasoning steps of several models such as *GPT-3.5-Turbo*, *DeepSeek-Chat*, *Qwen3:14B*, etc., which is due to the methods equipped by LLMs lacks of ability for retrospection. This case is not only observed in this work but also reported in prior works (Zhu et al., 2025; Feng et al., 2025; Tyen et al., 2024).

6.2 Impact of Increasing m

We select 3 LLMs, namely *DeepSeek-Reasoner*, *GPT-5-Mini*, and *Qwen-Turbo*, and evaluate them on the additional constructed datasets, where the number of events is fixed to $n = 2$ while the num-

[6]An example of *Temporal Semantics and Reasoning Misalignment* is demonstrated in Listing 4 in Appendix A.

[7]An example of *Context Hypothesis Detachment* is demonstrated in Listing 5 in Appendix A.

ber of formula operators m increases from 1 to 9, specifically $m \in \{1, 2, 3, 4, 5, 7, 9\}$. This aims to explore whether the performance of LLMs on TR challenges is stable as the number of formula operators increases and also whether the increase of the number of formula operators can introduce more complexity.

The results of the models equipped with each method are demonstrated in Figure 4 in Appendix A. While four methods exhibit an oscillation of accuracy while $m < 7$, the *Few-Shot CoT* method maintains a stable and robust performance. Furthermore, a sudden accuracy drop is observed at $m = 3$ for the other four methods. In particular, *Direct Prompting* shows an accuracy decrease at $m = 3$ followed by an increase up to $m = 7$, indicating poorer performance on simpler problems but improved performance on harder ones, which is counter-intuitive. Thus, to further investigate the reason, we conduct a qualitative analysis. We find that when $m <= 3$, the problems are relatively simple and the LLMs do not automatically and frequently invoke explicit reasoning to arrive at answers, in which while $m < 3$, the problems could be easy for LLMs to solve even without explicit reasoning, whereas while $m = 3$, the problems become hard for the LLMs and they still fail to invoke explicit reasoning process or they put less effort in reasoning process, leading to worse performance, which is a case that especially happens frequently in *Direct Prompting*. However, while $m > 3$, the LLMs invoke reasoning more frequently and take more effort in the reasoning process, resulting in the increase of accuracy.

In addition, while $m \geq 7$, the accuracy shows an obvious decreasing trend. From $m = 7$ to $m = 9$, grouped by methods across models, accuracy decreases by 6.44% in *Direct Prompting*, 7.44% in *Zero-Shot CoT*, 3.89% in *Few-Shot CoT*, 6.22% in *Self-Consistency*, 7.11% in *Least-to-Most*, while grouped by models across methods, accuracy decreases by 6.87% in *DeepSeek-Reasoner*, 7.53% in *GPT-5-Mini*, and 4.27% in *Qwen-Turbo*. Overall, increasing the number of formula operators from 7 to 9 results in an average accuracy drop of 6.22%, with a minimum drop of 0.3% and the maximum drop of 12%. This indicates that the increase of the number of formula operators can significantly introduce more complexity to the problems and also shows that the TR ability of LLMs lacks consistency and robustness as TR problems complexity grows due to involving more formula operators.

6.3 Impact of increasing n

In addition, the selected 3 models are evaluated on another additional constructed datasets, where the number of formula operators is fixed to $m = 2$ while the number of events n, where $n > 1$, increases from 2 to 9, specifically $n \in \{2, 3, 4, 5, 7, 9\}$. Similar to Section 6.2, this, however, aims to explore how the TR ability of LLMs is affected as the number of events increases and also whether the increase of the number of events can lead to more complexity.

As shown in Figure 5 in Appendix A, across all models and methods, they show a more consistent trend of accuracy decrease as the number of events increases. From $n = 2$ to $n = 9$, grouped by methods across models, accuracy drops by 16.22% in *Direct Prompting*, 18.78% in *Zero-Shot CoT*, 19.44% in *Few-Shot CoT*, 18% in *Self-Consistency*, and 17% in *Least-to-Most*, while grouped by models across methods, accuracy decreases by 19.80% in *DeepSeek-Reasoner*, 16.20% in *GPT-5-Mini*, and 17.67% in *Qwen-Turbo*. Overall, increasing the number of events from 2 to 9 results in an average accuracy drop of 17.89%, with a minimum drop of 11% and a maximum drop of 24.67%. This decline is more pronounced than that observed when increasing the number of formula operators, indicating that the increase of the number of events can considerably lead to more complexity of TR problems and that the TR ability of LLMs is unstable and not robust as the complexity of TR problems increases due to involving more events.

7 Conclusion

In this work, we evaluate TR ability of LLMs from the perspective of formal logical reasoning over temporal information and design a pipeline for TR challenges generation, by leveraging random graph generation, LTL formula, and the NuSMV model checker. Using this pipeline, we construct a dataset, LTLBench, consisting of 2000 TR challenges, and conduct intensive evaluations on 12 LLMs across 5 methods. The results demonstrate the limitations of LLMs in handling the TR problem. We further qualitatively analyze their reasoning processes and identify 3 main reasoning issues. In addition, with additional experiments, we demonstrate that their performance is unstable and not robust as the number of formula operators and events increases, and also show that the pipeline can synthesize TR challenges in various levels of complexity and size.

Limitations

Considering the difficulty of generated TR problems, we only include a subset of LTL operators in the TR problem generation pipeline but exclude several LTL operators such as U (Until) and R (Release). However, incorporating more LTL operators should be easy and straightforward, for which the researchers only need to add new LTL operators in Algorithm 1 and add corresponding natural language semantics in the process of natural language generation described in Section 4.4. The flexibility of the pipeline enables future work to evaluate the more complex TR ability of LLMs.

In addition, this work mainly leverages LTL to generate TR problems, of which the expression ability is limited at linear event sequences, lacking of expression ability to handle events in branching-time. Thus, a natural step of the future direction is to utilize Computation Tree Logic (CTL) or CTL* to further construct more complicated TR problems. For example, in CTL, we can use $AE\phi$ to express that for all event sequences, ϕ eventually holds. We believe integrating the CTL and CTL* into this pipeline could further explore the TR ability boundaries of LLMs.

References

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, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Himanshu Beniwal, Dishant Patel, Kowsik Nandagopan D, Hritik Ladia, Ankit Yadav, and Mayank Singh. 2024. Remember this event that year? assessing temporal information and understanding in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16239–16348, Miami, Florida, USA. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Luca Chittaro and Angelo Montanari. 2000. Temporal representation and reasoning in artificial intelligence: Issues and approaches. *Annals of Mathematics and Artificial Intelligence*, 28(1):47–106.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Haotian Wang, Ming Liu, and Bing Qin. 2023. Timebench: A comprehensive evaluation of temporal reasoning abilities in large language models. (arXiv:2311.17667). ArXiv:2311.17667 [cs].

A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. 2002. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, volume 2404 of *LNCS*, Copenhagen, Denmark. Springer.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011, Brussels, Belgium. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. 2024. Test of time: A benchmark for evaluating llms on temporal reasoning. *Preprint*, arXiv:2406.09170.

Yiyang Feng, Yichen Wang, Shaobo Cui, Boi Faltings, Mina Lee, and Jiawei Zhou. 2025. Unraveling misinformation propagation in LLM reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11683–11707, Suzhou, China. Association for Computational Linguistics.

Valentin Goranko and Antje Rumberg. 2024. Temporal Logic. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*, Summer 2024 edition. Metaphysics Research Lab, Stanford University.

Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, Philip S. Yu, and Zhijiang Guo. 2023. Do large language models know about facts? *Preprint*, arXiv:2310.05177.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. (arXiv:2310.06825). ArXiv:2310.06825 [cs].

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Saul A Kripke. 1963. Semantical considerations on modal logic. *Acta philosophica fennica*, 16.

Fred Kröger and Stephan Merz. 2008. *Temporal logic and state systems*. Texts in theoretical computer science: An EATCS series. Springer, Berlin.

Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. 2023. Grounding complex natural language commands for temporal tasks in unseen environments. (arXiv:2302.11649). ArXiv:2302.11649 [cs].

Zijia Liu, Peixuan Han, Haofei Yu, Haoru Li, and Jiaxuan You. 2025. Time-r1: Towards comprehensive temporal reasoning in llms. *Preprint*, arXiv:2505.13508.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Lianhui Qin, Aditya Gupta, Shyam Upadhyay, Luheng He, Yejin Choi, and Manaal Faruqui. 2021. Timedial: Temporal commonsense reasoning in dialog. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, page 7066–7076, Online. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Yoav Shoham and Nita Goyal. 1988. Chapter 11 - temporal reasoning in artificial intelligence. In Howard E. Shrobe and the American Association for Artificial Intelligence, editors, *Exploring Artificial Intelligence*, pages 419–438. Morgan Kaufmann.

Qingyu Tan, Hwee Tou Ng, and Lidong Bing. 2023. Towards benchmarking and improving the temporal reasoning capability of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 14820–14835, Toronto, Canada. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. 2024. 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. Association for Computational Linguistics.

Siddharth Vashishtha, Adam Poliak, Yash Kumar Lal, Benjamin Van Durme, and Aaron Steven White. 2020. Temporal reasoning in natural language inference. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, page 4070–4078, Online. Association for Computational Linguistics.

Lluis Vila. 1994. A survey on temporal reasoning in artificial intelligence. *Ai Communications*, 7(1):4–28.

Felix Virgo, Fei Cheng, and Sadao Kurohashi. Improving event duration question answering by leveraging existing temporal information extraction data.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yuqing Wang and Yun Zhao. 2024. Tram: Benchmarking temporal reasoning for large language models. (arXiv:2310.00835). ArXiv:2310.00835 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. 2024. Large language models can learn temporal reasoning. (arXiv:2401.06853). ArXiv:2401.06853 [cs].

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed H. Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *CoRR*, abs/2205.10625.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. 2025. Where llm agents fail and how they can learn from failures. *Preprint*, arXiv:2509.25370.

Weijun Zhu. 2021. Big data on linear temporal logic formulas. In *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, page 544–547, Chongqing, China. IEEE.

A Appendix

Prompt Template 1: Direct Prompting, in which *question* is the placeholder for TR problem.

{question}

The answer must be either 'True' or 'False'.

Prompt Template 2: Zero-Shot CoT, in which *question* is the placeholder for TR problem.

{question}

Let's think step by step.

First, let's identify what the hypothesis is asking.
Then, let's understand and examine each condition.
Let's trace and reason through the event sequences.
Finally, let's determine if the hypothesis can be satisfied.

Please reason through this problem thoroughly before answering.
The final answer must be either 'True' or 'False'.

Prompt Template 3: Few-Shot CoT, in which *question* is the placeholder for TR problem.

=== Example 1 ===

Context:
Initially, event3 happened. After event3, either event1, or event2 must happen. After event1, either event2, or event3 must happen. After event2, event3 must happen.

Hypothesis:
C1: Event1 happens or event3 happens.
C2: Event1 happens or C1 holds.
C3: C2 eventually holds.

Let's think step by step:
1. We need to check if C3 (C2 eventually holds) is true.
2. C2 is: Event1 happens or C1 holds
3. C1 is: Event1 happens or event3 happens
4. So C2 simplifies to: Event1 happens or (Event1 happens or event3 happens)
5. This is logically equivalent to: Event1 happens or event3 happens
6. The initial state has event3 = true
7. Therefore, C2 holds at the initial state (since event3 is true)
8. Since C2 holds at the initial state, C3 (C2 eventually holds) is True

Answer: True

=== Example 2 ===

Context:
Initially, event2 happened. After event2, event3 must happen. After event3, no other events can happen. After event1, either event2, or event3 must happen.

Hypothesis:
C1: Event2 happens and event3 happens.
C2: C1 holds in the next state.
C3: C2 eventually holds.

Let's think step by step:
1. We need to check if C3 (C2 eventually holds) is true.
2. C2 is: C1 holds in the next state, where C1 is (event2 AND event3)
3. For C1 to be true, both event2 and event3 must be true simultaneously
4. In any single state, only one event can be true at a time
5. Starting from event2, the next state must be event3
6. From event3, it stays at event3 (no other events can happen)
7. At no point can event2 and event3 be true simultaneously in the same state
8. Therefore, C1 is always false
9. If C1 is always false, then C2 (C1 in the next state) is always false
10. If C2 is always false, then C3 (C2 eventually holds) is False

Answer: False

=== Now solve this problem ===

""" + {question} + """

Let's think step by step.

First, let's identify what the hypothesis is asking.
Then, let's understand and examine each condition.
Let's trace and reason through the event sequences.
Finally, let's determine if the hypothesis can be satisfied.

Please reason through this problem thoroughly before answering.
The final answer must be either 'True' or 'False'.

Prompt Template 4: Least-to-Most Question Breakdown, in which *question* is the placeholder for TR problem.

{question}

Let me break this down into smaller, manageable parts:

1. What events are involved and what are the initial conditions?
2. What are the transition rules between events?
3. What does each sub-condition (C1, C2, C3, etc.) mean?
4. How do these conditions relate to each other?

Please answer each of these questions step by step.

Prompt Template 5: Least-to-Most Question Solving, in which *question* is the placeholder for TR problem and *breakdown_response* is the response of LLMs after prompted to breakdown a given question.

{question}

Based on the following breakdown of the problem:

{breakdown_response}

Now, let's use this understanding to solve the main problem step by step:

1. Start with the simplest conditions and work up to the complex ones
2. Trace through the event sequences
3. Check if the final hypothesis can be satisfied

The final answer must be either 'True' or 'False'.

```
=== Context ===

Initially, event1 happened. After event2, either event1, or event3 must happen. After event1, event3 must happen. After event3, event2 must happen. After event3, event2 must happen.

=== Hypothesis ===

C1: Event2 happens in the next state.
C2: C1 eventually holds.
C3: Event3 happens and C2 holds.

C3 is True or False?

=== Response ===

Let's analyze the problem step by step.

### Step 1: Understanding the Context
- Initially, **event1** happens.
- After **event2**, either **event1** or **event3** must happen.
- After **event1**, **event3** must happen.
- After **event3**, **event2** must happen.
- After **event3**, **event2** must happen (this is repeated).

### Step 2: Analyzing the Hypothesis
- **C1**: Event2 happens in the next state.
- **C2**: C1 eventually holds.
- **C3**: Event3 happens and C2 holds.

### Step 3: Tracing the Event Sequences
1. **Initial State**: event1 is true.
2. From event1, the next event must be event3 (as per the context).
3. Now, we have event3 true.
4. After event3, event2 must happen (as per the context).
5. Now, we have event2 true.

### Step 4: Evaluating C1, C2, and C3
- **C1**: Event2 happens in the next state.
```

```
  - In the initial state, event1 is true, and the next state will be event3. After event3, event2 will happen. Therefore, C1 will hold true in the next state after event3.

- **C2**: C1 eventually holds.
  - Since C1 holds true after event3, C2 is also true because C1 eventually holds.

- **C3**: Event3 happens and C2 holds.
  - Event3 does happen (as we established in the sequence), and C2 holds true (as we established that C1 eventually holds). Therefore, C3 is true.

### Conclusion
Since C3 is true (both conditions of C3 are satisfied), the final answer is:

**Answer: True**
```

Listing 4: An example of Temporal Semantics and Reasoning Misalignment.

```
=== Context ===

Initially, event2 happened. After event3, event1 must happen. After event1, event3 must happen. After event2, event1 must happen.

=== Hypothesis ===

C1: Event2 always happens.
C2: Event2 always happens.
C3: C1 holds and C2 holds.

C3 is True or False?

=== Response ===

The answer is: **True**

### Explanation:

- The hypothesis states:
  - **C1: Event2 always happens.**
  - **C2: Event2 always happens.**
  - **C3: C1 holds and C2 holds.**

Since **C1** and **C2** are **identical statements**, both asserting that "Event2 always happens," then **C3** (which says "C1 holds and C2 holds") is logically **true** because both conditions are the same and therefore both true.

Thus, **C3 is True**.
```

Listing 5: An example of Context Hypothesis Detachment.

Algorithm 1 Generate LTL Formulas

```
procedure GENERATELTLFORMULAS(states, formulaLength, formulaCount)
    Input: states - an array of atomic states representing events
    Input: formulaLength - the number of operators
    Input: formulaCount - the number of formulas to generate
    Output: an array of LTL formulas
    unaryOperators ← [X, G, F, !]
    binaryOperators ← [&, |, →]
    operators ← unaryOperators + binaryOperators
    B ← [[ ] of size formulaLength + 1 ]
    B[0] ← states
    formulas ← [ ]
    for i ← 1 to formulaCount do
        for j ← 1 to formulaLength do
            x ← randomly choose an operator from operators
            if x ∈ unaryOperators then
                y ← randomly sample a formula from B[j − 1]
                newFormula ← [x, y]
            else
                s ← randomly sample a integer from [0, j)
                y1 ← randomly sample a formula from B[s]
                y2 ← randomly sample a formula from B[j − 1 − s]
                newFormula ← [y1, x, y2]
            end if
            append newFormula to B[j]
        end for
        formula ← B[formulaLength][−1]
        append formula to formulas
    end for
    return formulas
end procedure
```

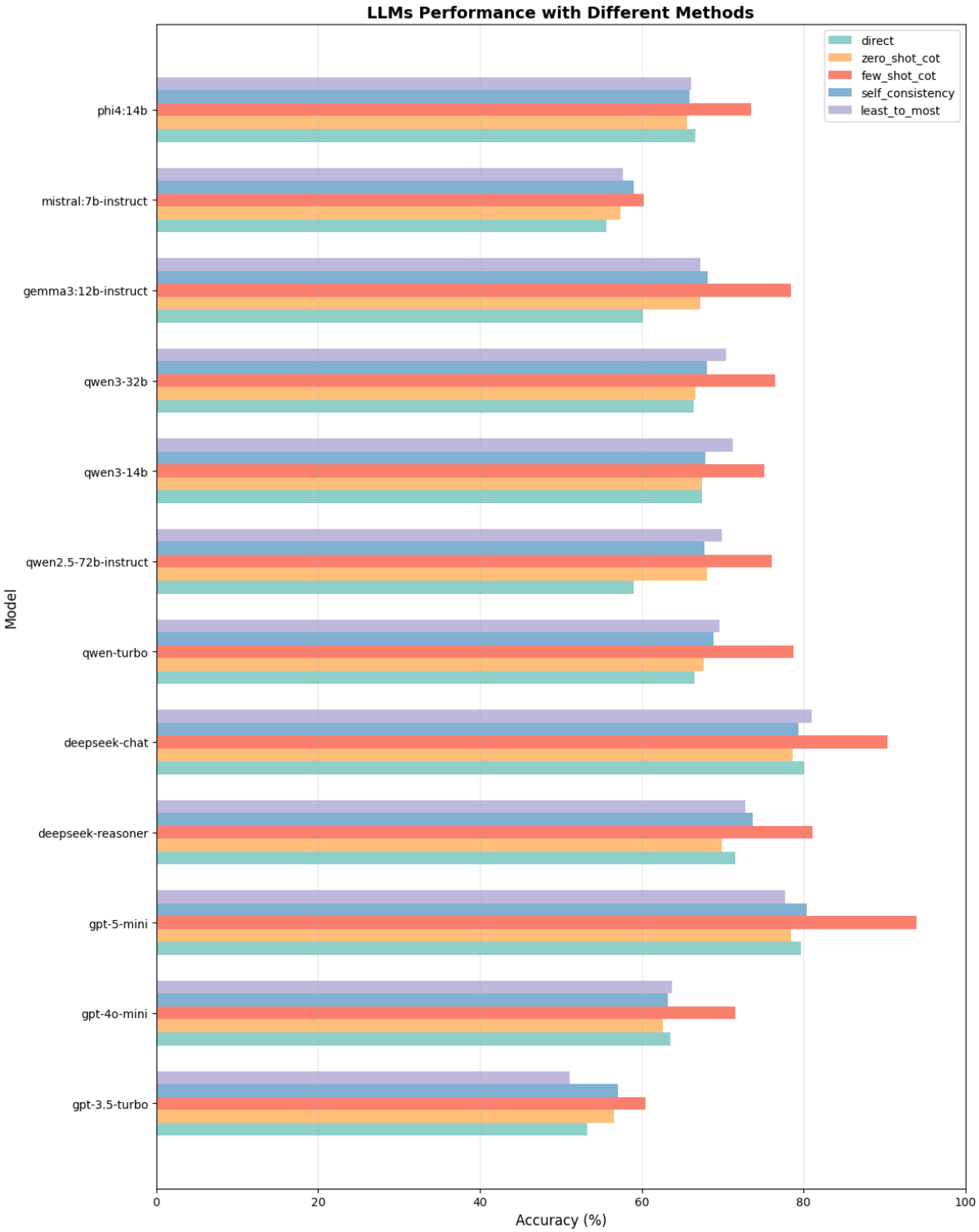
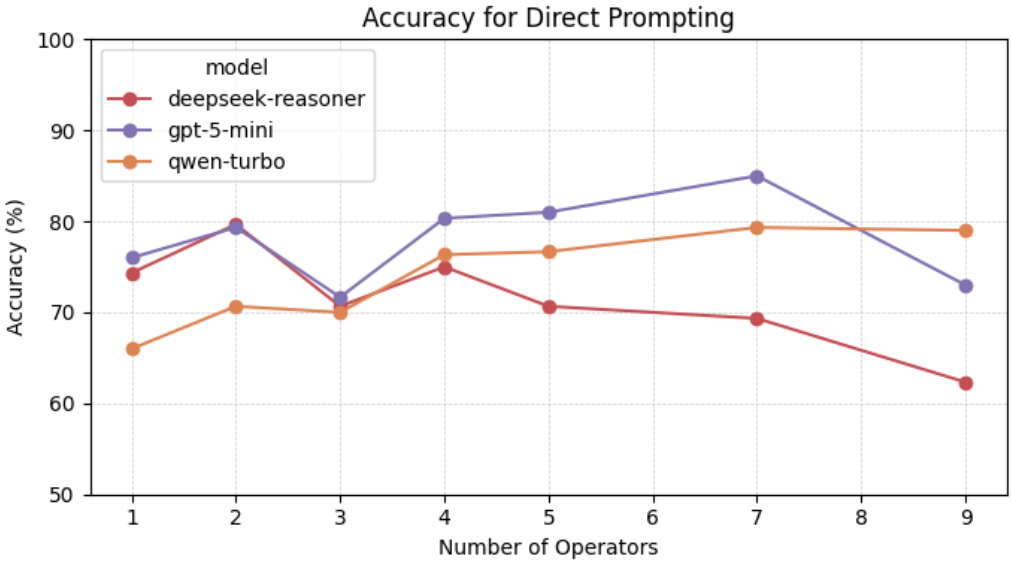
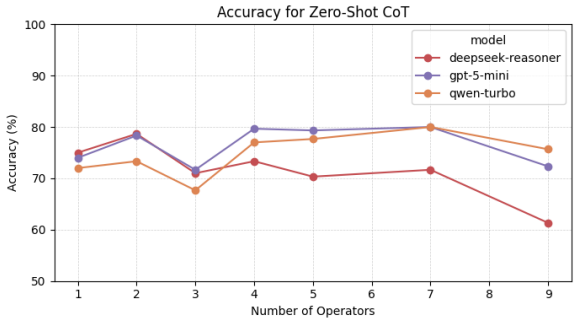


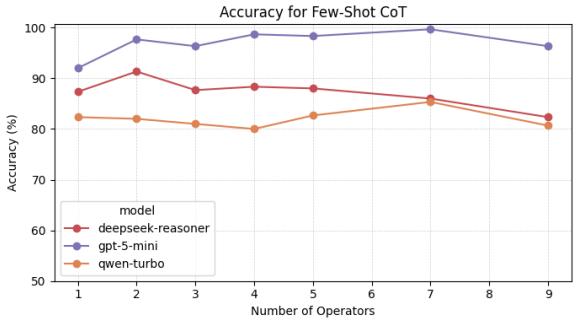
Figure 3: LLMs Performance with Different Methods.



(a) Direct Prompting

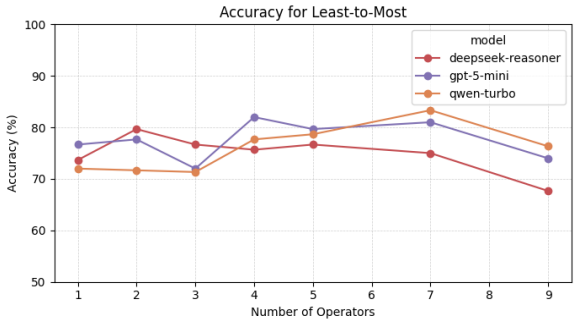


(b) Zero-Shot CoT



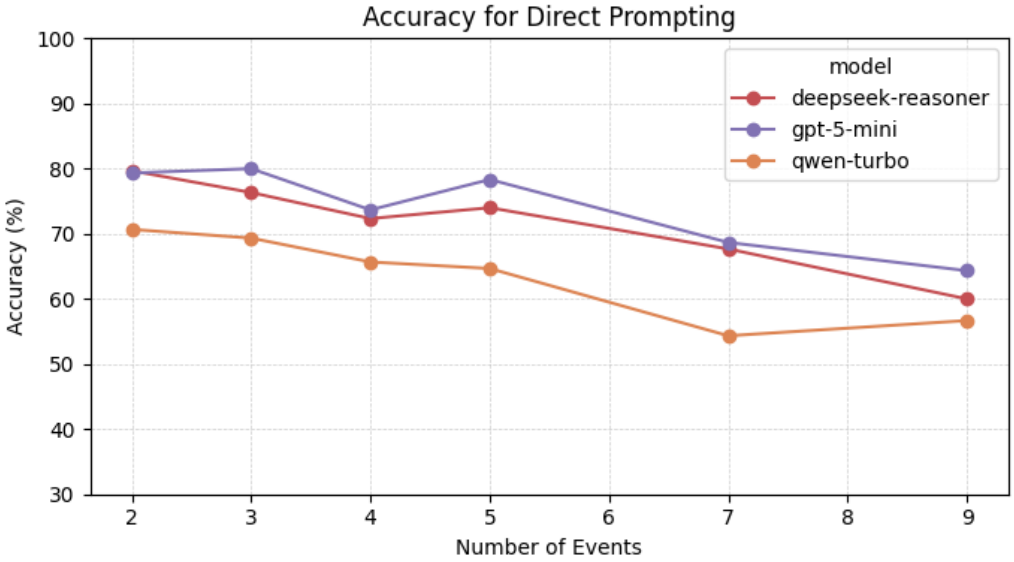
(c) Few-Shot CoT

(d) Self-Consistency

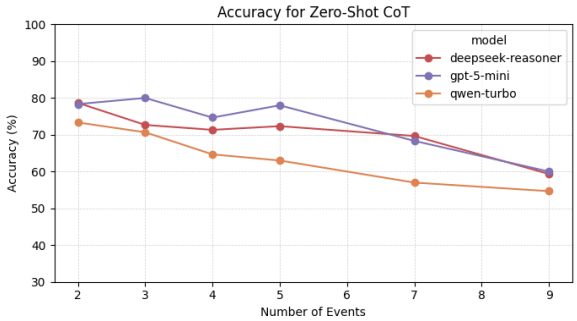


(e) Least-to-Most

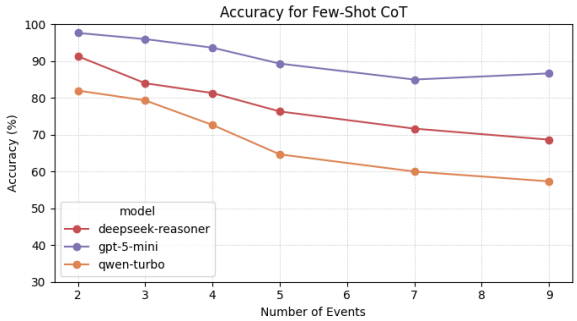
Figure 4: Accuracy for *DeepSeek-Reasoner*, *GPT-5-Mini*, and *Qwen-Turbo* equipped with each method as the number of formula operators increases.



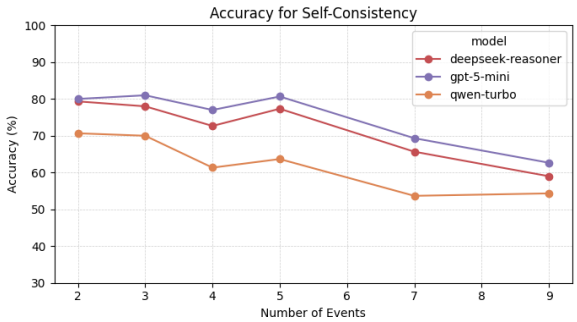
(a) Direct Prompting



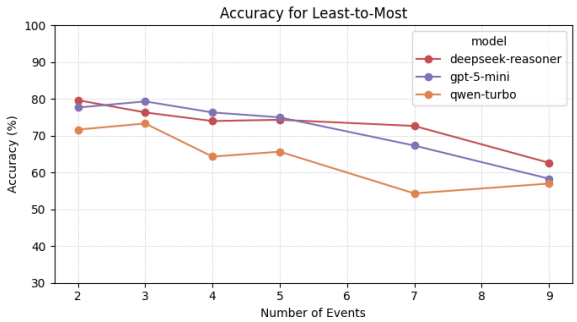
(b) Zero-Shot CoT



(c) Few-Shot CoT



(d) Self-Consistency



(e) Least-to-Most

Figure 5: Accuracy for *DeepSeek-Reasoner*, *GPT-5-Mini*, and *Qwen-Turbo* equipped with each method as the number of events increases.