RULEARENA: A BENCHMARK FOR LLM RULE-GUIDED REASONING IN REAL-WORLD SCENARIOS

Ruiwen Zhou[1] **Wenyue Hua**[1] **Liangming Pan**[2] **Sitao Cheng**[1]
Xiaobao Wu[1,3] **En Yu**[1] **William Yang Wang**[1]
[1]University of California, Santa Barbara [2]University of Arizona
[3]Nanyang Technological University

ABSTRACT

This paper introduces RULEARENA, a challenging benchmark to evaluate the ability of large language models (LLMs) to follow complex, real-world rules in reasoning. Covering three practical domains—airline baggage fees, NBA transactions, and tax regulations—RULEARENA assesses LLMs' proficiency in handling intricate natural language instructions that demand long-context understanding, logical reasoning, and accurate math computation. Two key attributes distinguish RULEARENA from traditional rule-based reasoning benchmarks: (1) it extends beyond standard first-order logic representations, and (2) it is grounded in authentic, practical scenarios, providing insights into the suitability and reliability of LLMs for real-world applications. Our findings reveal several notable limitations in LLMs: (1) they struggle to identify and apply the appropriate rules, frequently becoming confused by similar but distinct regulations, (2) they cannot consistently perform accurate mathematical computations, even when they correctly identify the relevant rules, and (3) in general, they perform poorly in the benchmark. We also observe a significant performance boost when LLMs are provided with external tools. These results highlight significant challenges and promising directions in advancing LLMs' rule-guided reasoning capabilities in real-life applications

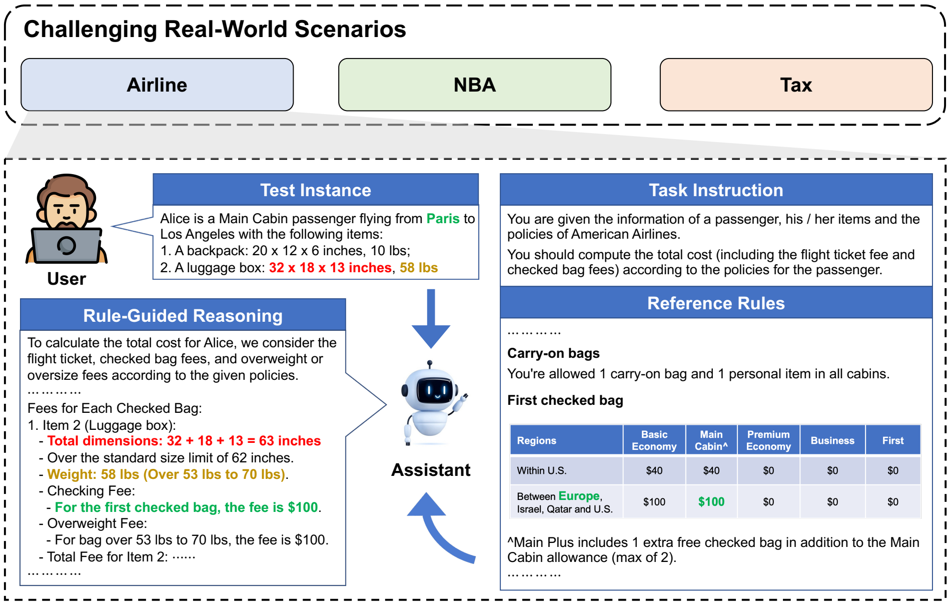
1 INTRODUCTION

Recently, Large Language Models (LLMs) (Touvron et al., 2023; OpenAI, 2023; Team, 2023; Anthropic, 2024) have demonstrated remarkable capabilities across many real-world applications. However, their limited domain-specific knowledge often leads to unfaithful or misleading output, which can cause significant risks and financial liabilities. For example, Canadian airline was recently required to compensate a customer who received incorrect guidance from the airline's chatbot[1]. These challenges highlight the need for robust, real-world benchmarks that assess how faithfully and accurately LLMs can follow real-life instructions and adhere to relevant regulations, thereby ensuring reliable and safe outputs for deployment.

Although several studies have examined LLMs' instruction-following abilities (Chen et al., 2024b; Jiang et al., 2024; Wen et al., 2024), they mainly focused on style constraints, such as the format (Zhou et al., 2023) or topic of responses. Yet, the significance of instruction-following extends well beyond style compliance. In many problem-solving scenarios, instructions function as *rules*: they impose logical constraints on the reasoning process and specify how answers should be derived from inputs. However, limited attention has been paid to LLMs' capacity to follow complex rules.

Existing research (Mu et al., 2023; Sun et al., 2024) largely addresses only single-step, first-order logic reasoning or artificially synthesized logical tasks. In contrast, real-world rules frequently appear in diverse and nuanced natural language forms. They may involve intricate logical structures, including the need for parallel reasoning across multiple rules or navigating interdependent rule sets. For instance, to calculate the fees for checked luggage when taking flights, one needs to consider the base price for checking each item, the overweight and oversize charges, and how these charges should be aggregated together. The extent to which LLMs can accurately follow these com-

[1]`https://www.theguardian.com/world/2024/feb/16/air-canada-chatbot-lawsuit`



Test Instance

Alice is a Main Cabin passenger flying from Paris to Los Angeles with the following items:
1. A backpack: 20 x 12 x 6 inches, 10 lbs;
2. A luggage box: 32 x 18 x 13 inches, 58 lbs

Task Instruction

You are given the information of a passenger, his / her items and the policies of American Airlines.
You should compute the total cost (including the flight ticket fee and checked bag fees) according to the policies for the passenger.

Reference Rules

… … … …

Carry-on bags

You're allowed 1 carry-on bag and 1 personal item in all cabins.

First checked bag

Regions	Basic Economy	Main Cabin^	Premium Economy	Business	First
Within U.S.	$40	$40	$0	$0	$0
Between Europe, Israel, Qatar and U.S.	$100	$100	$0	$0	$0

^Main Plus includes 1 extra free checked bag in addition to the Main Cabin allowance (max of 2).
… … … …

Rule-Guided Reasoning

To calculate the total cost for Alice, we consider the flight ticket, checked bag fees, and overweight or oversize fees according to the given policies.
… … … …
Fees for Each Checked Bag:
1. Item 2 (Luggage box):
- Total dimensions: 32 + 18 + 13 = 63 inches
- Over the standard size limit of 62 inches.
- Weight: 58 lbs (Over 53 lbs to 70 lbs).
- Checking Fee:
  - For the first checked bag, the fee is $100.
- Overweight Fee:
  - For bag over 53 lbs to 70 lbs, the fee is $100.
- Total Fee for Item 2: ……
… … … …

Figure 1: Overview of RULEARENA. RULEARENA contains 95 commonly used and moderately complex rules and 816 test problems from three representative real-world scenarios - airline luggage fees, NBA transactions, and taxation policies. LLMs are given a the task instruction, the reference rules in this scenario, and a user instance, and required to conduct reasoning and computation for the user input under the guidance of reference rules.

plex, real-world rules—an ability we term *rule-guided reasoning*—remains unknown. To better understand the complexity and practical implications of rule-guiding reasoning, we introduce a new evaluation benchmark, RULEARENA, grounded in realistic scenarios.

As illustrated in Figure 1, RULEARENA is developed from three representative, real-world scenarios: (1) airline luggage fee policies, (2) NBA transactions, and (3) taxation policies. From these domains, we collect authentic rules currently implemented by companies or government agencies. For each domain, we construct a set of challenging test problems, pairing each question with a ground-truth solution, and then evaluate a range of state-of-the-art LLMs on their ability to conform to the rules. LLMs are provided with the domain-specific task instructions and reference rules, and required to resolve each test problem through reasoning according to the question and reference rules.

Our contributions in this work can be summarized in three main points:

- **A diverse collection of real-world rules:** We assemble a comprehensive set of 95 policies/rules drawn from these three real-world scenarios.
- **A challenging benchmark and novel evaluation metrics:** Using the collected rules, we introduce RULEARENA, a new benchmark containing 816 datapoints designed to test LLMs' rule-guided reasoning ability. We further propose a suite of evaluation metrics for both rule selection and rule application, providing fine-grained insights into LLMs' performance.
- **A comprehensive analysis of prevalent challenges:** By examining common failure cases and identifying difficult rule types, we uncover several systematic issues that limit current LLMs' rule-guided reasoning capabilities and promising directions for improvement.

2 RELATED WORK

Complex Instruction-following Benchmarks A wide range of benchmarks has been designed to evaluate LLMs' instruction-following abilities from various perspectives, including semantics (Zheng et al., 2023; Li et al., 2023; Liu et al., 2023; Wu et al., 2024a;b), format (Xia et al., 2024; Tang et al., 2023), and response length (Chen et al., 2024a; Sun et al., 2023). To further probe complexity, some works have introduced benchmarks that construct complex instructions through compositional methods. For example, **WizardLM** (Xu et al., 2023) generates intricate tasks by combining simpler instructions, while **CELLO** (He et al., 2024) uses task descriptions and input texts to create complex prompts grounded in real-world scenarios. **ComplexBench** (Wen et al., 2024) adopts multiple compositional structures to integrate atomic requirements into more challenging instructions. In contrast, our work focuses on instructions derived directly from real-life scenarios, where naturally occurring complexities arise from multifaceted constraints incurred by inputs on the set of instructions.

Logical Reasoning Benchmarks Extensive research has explored benchmarks for mathematical (Koncel-Kedziorski et al., 2016; Ling et al., 2017; Amini et al., 2019; Cobbe et al., 2021; Hendrycks et al., 2021) and logical (Mao et al., 2019; Gupta et al., 2020; Tafjord et al., 2021; Zhong et al., 2022; Han et al., 2022; Zhang & Ding, 2024) reasoning, evaluating LLMs' abilities to solve math problems of varying difficulty, tackle coding challenges, and engage in deductive logic. Although these benchmarks test models' reasoning skills, their logical constraints are often represented in simplified, formal systems, such as propositional (Hua et al., 2024) or first-order logic (Zhu et al., 2023; Mu et al., 2023; Sun et al., 2024). In contrast, our benchmark deals with rules that arise in natural language, capturing a richer, more realistic set of constraints. Such natural language rules extend beyond neatly formalized logical representations, often express higher-order logic and more intricate relationships than typical propositional or first-order logic formalizations.

3 RULEARENA

In this section, we present the RULEARENA benchmark and its construction process. We begin by describing the domains we have chosen and the corresponding regulations from which our rules are collected. We then describe how problems with varying difficulty levels are generated and how the ground-truth solutions are computed. Finally, we present the evaluation metrics we used to evaluate whether correct rules are correctly applied.

3.1 DOMAINS AND RULE COLLECTION

We select three real-life domains both familiar in everyday life and have a high level of complexity:

Airline. It requires LLM to calculate the total cost for one or more passengers, including their flight ticket and checked bag fees. The regulations are extracted from policy of American Airlines[2]. The complexity stems from the fact that baggage costs vary according to factors such as cabin class, flight route, the number of checked bags, and the size of each bag. Consequently, LLMs must carefully identify the correct relevant rules and apply them accurately to calculate the final cost.

NBA transaction. It requires LLMs to determine whether one or more specified transactions are allowed. The regulations are extracted from the *2023 NBA Collective Bargaining Agreements*[3] (CBA) and excerpt from the *NBA Constitution and By-Laws*[4]. Complexity arises from the numerous factors influencing transaction eligibility, including the player's contract value, salary-matching constraints, and the specific transaction date. LLMs must accurately identify and apply the relevant rules from the agreement to determine whether a given transaction can proceed.

Tax. It requires LLMs to calculate the income tax for a person or family given their financial information. The regulations are collected from Internal Revenue Service[5]. Although taxes are a common and universally encountered aspect of modern life, they are also known for their complexity. This complexity stems from a wide range of factors, including salary income, investment gains, home ownership and related expenses, etc. To arrive at the correct tax amount, LLMs must navigate and apply the appropriate rules drawn from these multifaceted conditions.

The statistics for the collected rules are summarized in Table 1. Although the total number of rules is relatively small, each rule averages just under 400 tokens in length, tokenized by Llama-3.1 tokenizer. This presents a substantial challenge for both rule comprehension and the handling of long contexts. For more details on how we collect our rules, please refer to Appendix B.1.

[2]https://www.aa.com/i18n/customer-service/support/optional-service-fees.jsp

[3]https://ak-static.cms.nba.com/wp-content/uploads/sites/4/2023/06/2023-NBA-Collective-Bargaining-Agreement.pdf

[4]https://ak-static-int.nba.com/wp-content/uploads/sites/3/2015/12/NBA-Constitution-and-By-Laws.pdf

[5]https://www.irs.gov/forms-instructions

	Airline	NBA	Tax
# Rules	10	54	31
Average # Tokens	376	398	359

Table 1: Statistics of rules in each domain.

3.2 Problem Annotation

After gathering the relevant rules for each domain, we construct challenging test problems designed to evaluate whether LLMs can produce correct outputs from the provided rules.

Airline. The problems are generated by randomly selecting passenger information (e.g., cabin class, itinerary, ticket price) and their checked baggage details (e.g., dimensions, weight). We convert each regulation into a corresponding rule-based script, enabling the direct calculation of ground-truth answers by executing these scripts. LLM performance is then assessed by comparing the model's computed solutions to the script-derived ground truths, step by step.

NBA Transaction. The problems consist of proposed trades that may or may not comply with NBA regulations. As these problems require a wide variety of operations and rule sets, fully automated generation and evaluation are difficult. Therefore, we employ annotators familiar with NBA transaction rules to curate complex test cases and identify all the relevant rules needed to resolve each case (further details in Appendix B.2). For each problem, we ask the LLM whether the transaction is legit or not based on the regulations. If LLM thinks the transaction is legit, it should generate "Yes"; otherwise, it needs to identify the specific team and transaction that violates the rules.

Tax. The problems are randomly generated from hypothetical taxpayer profiles including information such as income levels, filing status, *etc*. IRS tax regulations are translated into rule-based scripts to compute ground-truth tax obligations. As with the airline scenario, we measure LLM accuracy by comparing the model's step-by-step calculations with those derived directly from the scripts.

3.3 Difficulty control

To assess LLMs' capabilities under varying levels of complexity, we create problems with different degrees of difficulty. We define three levels of difficulties in each domain:

Airline. The difficulty is controlled by adjusting the number of bags a passenger carries.

NBA Transaction. The difficulty is determined by adjusting the number of teams, players, and transactions involved in a scenario.

Tax. The difficulty is raised by gradually introducing additional tax forms and thus relevant rules.

3.4 Evaluation Metrics

To achieve a comprehensive evaluation of the rule-following abilities of Large Language Models (LLMs), we introduce a set of evaluation metrics. Unlike existing benchmarks (Hua et al., 2024; Fan et al., 2023; Zhu et al., 2023), which primarily rely on simple metrics such as answer accuracy or BLEU scores, our approach aims to conduct a more detailed analysis of the step-by-step rule-guided reasoning process. This analysis includes examining each rule application to determine whether the rule should be applied, whether any rules are missed, and whether the rule application computation process is accurate.

For each domain, assuming a set $\mathcal{T}$ of N problems and a set $\mathcal{R}$ of M. For each problem $t_i = (q_i, a_i, R_i)$, we have a query q_i, an answer a_i, and a set of relevant rules R_i, together with a rule-usage matrix $U \in \mathbb{R}^{N\times M}$, where each item $U_{i,r} \in \{0,1\}$ indicates whether a rule r is used by an LLM in problem t_i. Matrix U can be approximately obtained by parsing LLMs' responses using an LLM, which we will introduce in Section 4.1.

Now we introduce two groups of metrics:

The first category focuses on problem-level evaluations: for each problem, we examine whether all necessary rules were applied, whether any extraneous rules were applied, and whether the final answer aligns with the ground-truth solution:

Problem-wise Recall: denoted as $\mathrm{R}(t_i)$, measures whether LLMs apply all relevant rules for a problem t_i, which is calculated as the proportion of relevant rules that are applied by LLMs:

$$\mathrm{R}(t_i) = \frac{\sum_{r \in R_i} \mathbb{I}(U_{i,r} = 1)}{\sum_r \mathbb{I}(r \in R_i)} \tag{1}$$

Problem-wise Rule Application Correctness: denoted as $\mathrm{AC}(t_i)$, measures whether LLMs apply rules correctly for a problem t_i, which is calculated as the proportion of correctly applied rules:

$$\mathrm{AC}(t_i) = \frac{\sum_{U_{i,r}=1} \mathbb{I}(r \text{ is correctly applied})}{\sum_r \mathbb{I}(U_{i,r} = 1)} \tag{2}$$

Problem-wise Precision: denoted as $\mathrm{P}(t_i)$, measures whether LLMs apply only relevant rules for a problem t_i, which is calculated as the proportion of applied rules that are relevant:

$$\mathrm{P}(t_i) = \frac{\sum_{U_{i,r}=1} \mathbb{I}(r \in R_i)}{\sum_r \mathbb{I}(U_{i,r} = 1)} \tag{3}$$

Problem-wise Accuracy: denoted as $\mathrm{Acc}(t_i)$, measures whether LLMs accurately answer the problem t_i comparing with ground-truth result. Assume the LLM provides answer $\tilde{a}_i$ for a problem t_i, the accuracy should be calculated as:

$$\mathrm{Acc}(t_i) = \mathbb{I}(\tilde{a}_i = a_i) \tag{4}$$

The second category of metrics focuses on rules rather than problems. For each rule in the domain, we assess whether it is applied to all problems that require it, and whether the problems it is applied to are exactly those that necessitate it:

Rule-wise Recall, denoted as $\mathrm{R}(r)$, measures whether LLMs apply r when r is relevant:

$$\mathrm{R}(r) = \frac{\sum_i \mathbb{I}(r \in R_i)\mathbb{I}(U_{i,r} = 1)}{\sum_i \mathbb{I}(r \in R_i)} \tag{5}$$

Rule-wise Rule Application Correctness, denoted as $\mathrm{AC}(r)$, measures whether LLMs correctly apply r:

$$\mathrm{AC}(r) = \frac{\sum_i \mathbb{I}(U_{i,r} = 1)\mathbb{I}(r \text{ is correctly applied})}{\sum_i \mathbb{I}(U_{i,r} = 1)} \tag{6}$$

Rule-wise Precision, denoted as $\mathrm{P}(r)$, measures whether r is relevant when LLMs apply r:

$$\mathrm{P}(r) = \frac{\sum_i \mathbb{I}(U_{i,r} = 1)\mathbb{I}(r \in R_i)}{\sum_i \mathbb{I}(U_{i,r} = 1)} \tag{7}$$

Problem-Wise Recall and *Problem-Wise Accuracy* are applied to all three domains to measure the ability of LLMs to match and aggregate rules and to comprehensively follow the rules. *Problem-Wise Application Correctness* is used on Airline and Tax tasks to evaluate whether LLMs can operate correctly under the guidance of rules, as these two tasks have clear procedures without ambiguous rules, while *Problem-Wise Precision* is used on NBA tasks to examine whether LLMs can differentiate similar rules applicable to different situations.

4 EXPERIMENTS

This section presents the experiments on benchmark. We first introduce the LLMs and prompting strategies we use to evaluate, and then present the evaluation result.

Models	Settings	Level 1				Level 2				Level 3			
		$\mathrm{P}(t)$	$\mathrm{AC}(t)$	$\mathrm{R}(t)$	$\mathrm{Acc}(t)$	$\mathrm{P}(t)$	$\mathrm{AC}(t)$	$\mathrm{R}(t)$	$\mathrm{Acc}(t)$	$\mathrm{P}(t)$	$\mathrm{AC}(t)$	$\mathrm{R}(t)$	$\mathrm{Acc}(t)$
Airline													
Llama-3.1 70B	0-shot	1.000	0.764	0.558	0.01	1.000	0.732	0.535	0.01	1.000	0.752	0.578	0.00
	1-shot	1.000	0.809	0.787	0.17	1.000	0.827	0.801	0.07	1.000	0.769	0.815	0.01
Qwen-2.5 72B	0-shot	1.000	0.636	0.586	0.01	1.000	0.627	0.554	0.01	1.000	0.588	0.544	0.00
	1-shot	1.000	0.836	0.908	0.19	1.000	0.818	0.901	0.10	1.000	0.801	0.904	0.01
Llama-3.1 405B	0-shot	1.000	0.854	0.604	0.03	1.000	0.844	0.587	0.06	1.000	0.845	0.570	0.01
	1-shot	1.000	0.919	**0.921**	**0.32**	1.000	0.897	**0.905**	0.16	1.000	0.870	**0.946**	0.04
Claude-3.5 Sonnet	0-shot	1.000	0.930	0.702	0.04	1.000	0.876	0.669	0.00	1.000	0.888	0.646	0.01
	1-shot	1.000	**0.960**	0.871	0.29	1.000	**0.966**	0.822	**0.30**	1.000	**0.972**	0.718	**0.11**
GPT-4o	0-shot	1.000	0.862	0.616	0.02	1.000	0.868	0.578	0.00	1.000	0.813	0.548	0.00
	1-shot	1.000	0.922	0.885	**0.32**	1.000	0.875	0.853	0.16	1.000	0.835	0.798	0.05
NBA Transaction													
Llama-3.1 70B	0-shot	0.579	–	0.428	0.40	0.498	–	0.246	0.36	0.540	–	0.250	0.22
	1-shot	0.560	–	0.565	0.49	0.466	–	0.386	0.25	0.578	–	0.438	0.26
Qwen-2.5 72B	0-shot	0.556	–	0.409	0.44	0.537	–	0.339	0.43	0.592	–	0.305	**0.30**
	1-shot	0.595	–	0.526	0.53	0.495	–	0.378	0.35	0.574	–	0.327	0.17
Llama-3.1 405B	0-shot	0.581	–	0.419	0.49	0.577	–	0.323	0.30	0.561	–	0.297	0.28
	1-shot	0.608	–	**0.550**	0.56	0.559	–	**0.439**	0.29	0.575	–	**0.461**	0.10
Claude-3.5 Sonnet	0-shot	0.660	–	0.457	0.38	0.630	–	0.373	0.40	0.588	–	0.292	0.28
	1-shot	**0.676**	–	0.528	**0.58**	**0.676**	–	0.410	**0.47**	**0.650**	–	0.371	0.26
GPT-4o	0-shot	0.650	–	0.446	0.40	0.570	–	0.327	0.26	0.603	–	0.291	0.24
	1-shot	0.616	–	0.506	0.40	0.597	–	0.392	0.28	0.569	–	0.318	0.20
Tax													
Llama-3.1 70B	0-shot	1.000	0.834	0.989	0.01	1.000	0.767	0.918	0.00	1.000	0.745	0.852	0.00
	1-shot	1.000	0.923	0.998	0.11	1.000	0.895	0.941	0.00	1.000	0.873	0.910	0.00
Qwen-2.5 72B	0-shot	1.000	0.888	0.998	0.10	1.000	0.835	0.944	0.01	1.000	0.785	0.903	0.00
	1-shot	1.000	0.931	**1.000**	0.17	1.000	0.919	0.934	0.00	1.000	0.921	0.868	0.00
Llama-3.1 405B	0-shot	1.000	0.923	0.999	0.16	1.000	0.876	0.964	0.02	1.000	0.797	**0.926**	0.00
	1-shot	1.000	0.941	**1.000**	0.24	1.000	0.914	0.958	0.03	1.000	0.873	0.880	0.00
Claude-3.5 Sonnet	0-shot	1.000	0.964	**1.000**	0.32	1.000	0.934	0.940	0.02	1.000	0.887	0.866	0.00
	1-shot	1.000	**0.979**	**1.000**	**0.64**	1.000	0.954	**0.969**	**0.16**	1.000	0.895	0.888	0.00
GPT-4o	0-shot	1.000	0.965	**1.000**	0.42	1.000	0.951	0.957	0.07	1.000	0.945	0.908	0.00
	1-shot	1.000	0.975	**1.000**	0.57	1.000	**0.975**	0.944	0.07	1.000	**0.982**	0.893	0.00

Table 2: Main problem-wise evaluation results on airline, NBA, and tax domains. $\mathrm{P}(t)$ denotes problem-wise precision, $\mathrm{AC}(t)$ denotes problem-wise rule application correctness, and $\mathrm{R}(t)$ denotes problem-wise recall.

4.1 Experiment Settings

LLMs. Our *rules*, which are prompted directly into LLMs, can be of a length up to 20,000 tokens. Therefore, we only consider LLMs that can handle such long contexts, including Llama-3.1 70B, Llama-3.1 405B (Dubey et al., 2024), Qwen-2.5 72B (Qwen Team, 2024), Claude-3.5 Sonnet (Anthropic, 2024), and GPT-4o (OpenAI, 2024) [6].

Prompting Strategies. Since rule-guided reasoning can be an intricate multi-step reasoning process in our three real-world scenarios, we use Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) reasoning by default. To further study if LLMs can learn to follow hard rules through in-context examples, we also compare 0-shot with 1-shot CoT given an example including a task of the lowest difficulty and its solution. Due to context limit, we do not further increase the number of in-context examples.

Output Parsing. To obtain the rule-usage matrix U we mentioned in Section 3.4, we utilize the structured output mode of GPT-4o (OpenAI, 2024) to parse the raw textual responses from LLMs. Specifically, for airline and tax problems we structuralize the ground-truth calculation process and ask GPT-4o to fill in problem-specific information according to an LLM's response, while for NBA problems we enumerate a list of all rules and ask GPT-4o to directly judge whether a specific rule is applied in an LLM's response. For details we refer our readers to Appendix C.

[6] Currently we have limited access to o1 API and thus only partial result, and we will update o1 performance once we have full access to the API

4.2 Main Results

This section provides a comprehensive analysis of benchmark results. The analysis is divided into two parts: problem-wise analysis and rule-wise analysis. The problem-wise analysis evaluates the performance of LLMs across different problems and difficulty levels; the rule-wise analysis delves into how effectively LLMs identify and apply specific rules, highlighting common failure modes and the impact of rule complexity and similarity.

4.2.1 Problem-wise Analysis

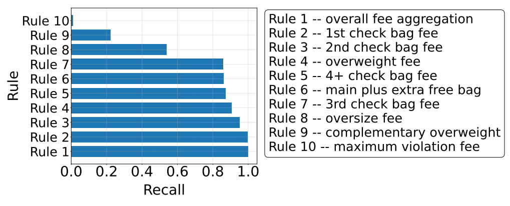
Table 2 presents the evaluation results[7]. Notice that the values of precision $(\mathrm{P})(t)$, rule application $(\mathrm{AC}(t))$, and recall $(\mathrm{R}(t))$ are much higher than accuracy $(\mathrm{Acc}(t))$. This is because solving a problem requires using multiple rules, hence one correct rule recall or application is insufficient for a correct answer. For example, if a problem requires 10 rules and only one rule is missed, $\mathrm{R}(t)$ is high as 0.9 while very probably leading to mistaken final answer $(\mathrm{Acc}(t) = 0)$.

Low Accuracy. Overall performance in problem result accuracy $(\mathrm{Acc}(t))$, as summarized in Table 2, remains unsatisfactory across all three scenarios. Under the 0-shot setting, even advanced models such as Llama 405B, Claude-3.5, and GPT-4o fail to produce correct answers for the simplest test problems. For more challenging problems, particularly in the airline and tax domains, $\mathrm{Acc}(t)$ rarely exceeds 10%. In 1-shot setting, we notice marked improvements on the easiest problems, yet the gains diminish as problem difficulty increases. These persistently low $\mathrm{Acc}(t)$ highlight the inherent complexity of the RuleArena benchmark and emphasize the need for more robust LLM reasoning and rule-following capabilities.

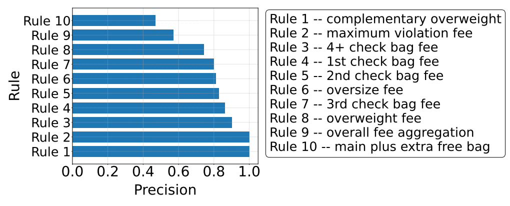
High Precision. In both the airline and tax scenarios, LLMs achieve 100% precision $(\mathrm{P}(t))$ in rule selection precision, consistently applying only those rules that are required. We notice that high $\mathrm{P}(t)$ stems from the relative clarity of the rules in these domains; the rules are neither highly similar nor ambiguous, making it straightforward to determine which ones apply. In contrast, the NBA scenario presents a more challenging environment, leading to noticeably lower $\mathrm{P}(t)$ in rule selection.

Low Recall. Despite exhibiting high $\mathrm{P}(t)$ in certain domains, LLMs often struggle with rule *recall* $(\mathrm{R}(t))$. Low $\mathrm{R}(t)$ in the airline, NBA, and more complex tax problems indicate that models do not fully grasp the reasoning workflows required. Consequently, they frequently fail to recall all necessary rules, reflecting an incomplete or superficial understanding of the underlying logic.

High Rule Application Correctness. While LLMs demonstrate relatively application correctness $(\mathrm{AC}(t))$ on rule application computation on average, $\mathrm{AC}(t)$ never reaches a perfect 100%. Occasional errors in mathematical calculations or logical operations emerge even under explicit rule guidance. Although these mistakes are not pervasive, a single computational error can significantly compromise the final output's accuracy $(\mathrm{Acc}(t))$ in many cases. This observation underscores the importance of improving the reliability of math computation abilities in LLMs.



(a) Recall



(b) Correctness

Figure 2: Rule-wise metrics of rules in airline domain.

[7]In NBA domain, the problem-wise correctness $(\mathrm{AC}(t))$ of rule application could not be computed due to the absence of step-by-step computation annotations. Generating such detailed annotations would require extensive human effort.

4.2.2 RULE-WISE ANALYSIS

Here, we provide a detailed examination of the rule-level evaluation. Figure 2 presents recall ($\mathrm{R}(r)$) and application correctness ($\mathrm{AC}(r)$) in airline domain and metrics for NBA transaction and tax domains are presented in Figure 4 and Figure 5 in Appendix. Table 3 summarizes the metric results by reporting the mean and variance of three key metrics across all rules: recall ($\mathrm{R}(r)$), application correctness ($\mathrm{AC}(r)$), and precision ($\mathrm{P}(r)$). The low variance observed in metrics such as $\mathrm{P}(r)$ within the airline and tax domains suggests that certain performance aspects are largely independent of the specific rules being applied. In contrast, the high variance seen in metrics like $\mathrm{R}(r)$ implies that recall performance is significantly influenced by the particular rules in question. Detailed analysis is presented below.

	Airline	NBA	Tax
$\mathrm{Mean}(\mathrm{P}(r))$	1.000	0.504	1.000
$\mathrm{Var}(\mathrm{P}(r))$	0.000	0.110	0.000
$\mathrm{Mean}(\mathrm{Ac}(r))$	0.798	–	0.828
$\mathrm{Var}(\mathrm{Ac}(r))$	0.026	–	0.047
$\mathrm{Mean}(\mathrm{R}(r))$	0.721	0.308	0.900
$\mathrm{Var}(\mathrm{R}(r))$	0.109	0.082	0.050

Table 3: Statistics of our three rule-wise metrics.

Rules with Low Recall. Certain rules are systematically overlooked across multiple data points, indicating that their neglect is not random but concentrated on specific rules. To understand which rules are most frequently disregarded, we identify the top-5 rules with the lowest *recall* ($\mathrm{R}(r)$), as presented in Table 4. We find that most of these rules are "non-essential," meaning they apply only under specific conditions. In contrast, "essential" rules must be applied in every scenario. For example, in the airline domain, essential rules define the baseline costs for each piece of luggage and the flight itself, making them relevant to all situations. Conversely, rules pertaining to overweight or oversized baggage only apply when such conditions arise, rendering them non-essential. Our observations indicate that these scenario-dependent, non-essential rules are more frequently neglected.

Airline		NBA		Tax	
Rule	essential	Rule	essential	Rule	essential
maximum violation fee		salary space consumption of bird right		education credits	
complementary overweight		salary space consumption of early bird right		american opportunity credit	
oversize fee		sign and trade maximum salary	✓	net profit	
3rd base check fee	✓	Arenas provision		ctc or other dependent credit	
main plus extra free bag		over 38 rule		taxes with qualified dividends	✓

Table 4: Top-5 rules of the lowest *recall* in ascent order of *recall*.

Rule with Low Application Correctness. We also identified the top-5 rules with the lowest *correctness* ($\mathrm{AC}(r)$), listed in Table 5. The majority of these rules are "compositional" in nature, requiring the aggregation of at least two previously computed intermediate results. By contrast, "non-compositional" rules demand at most a single mathematical operation involving a single intermediate result. Our analysis shows that compositional rules yield significantly lower $\mathrm{AC}(r)$ scores, indicating that LLMs struggle more with problems involving multiple reasoning steps than with straightforward, one-step computations.

Airline		Tax	
Rule	Composition	Rule	Composition
main plus extra free bag	✓	taxes with qualified dividends	✓
overall fee aggregation	✓	standard taxes	
overweight fee matching		itemized deductions	✓
3rd base check fee		standard deductions	✓
oversize fee matching	✓	total income	✓

Table 5: Top-5 rules of the lowest *correctness* in ascent order of *correctness*.

Rules with Low Precision. In both the airline and tax scenarios, all rules exhibit high *precision* ($\mathrm{P}(r)$), indicating that LLMs rarely apply irrelevant rules during the reasoning process. However, the NBA domain presents a different challenge, where multiple rules appear similar. As shown in Table 6, rules with low precision in the NBA domain usually have alternatives applicable under different conditions in the same situation. This pattern suggests that when rules are easily confused with one another, LLMs struggle to consistently identify and apply the correct one.

Rule	Substitutable
higher max criterion	
non bird right	✓
taxpayer mid level exception hard cap	✓
standard traded player exception	✓
salary increase ratio except bird right	✓

Table 6: Top-5 rules of the lowest *precision* in ascent order of *precision*.

4.3 In-Depth Analyses

4.3.1 What Impacts Rule Following?

We study the factors influencing LLM performance, as measured by $\mathrm{Acc}(t)$, and provide the complete experiment results in Appendix D.2. The main findings are:

Correlations between accuracy and other three metrics differ a lot. We compare the correlation between problem-wise metrics (i.e., $\mathrm{P}(t)$, $\mathrm{AC}(t)$, $\mathrm{R}(t)$) and accuracy $\mathrm{Acc}(t)$. The correlation is the most obvious and almost linear between $\mathrm{R}(t)$ and $\mathrm{Acc}(t)$, while highly non-linear or unclear between other two metrics and $\mathrm{Acc}(t)$.

In-context examples do not always help. We observe that LLMs generally provide better performances given 1-shot example on airline, tax, and (easy) NBA problems. However, when tackling more challenging NBA problems (Levels 2 and 3), providing an example increases $\mathrm{P}(t)$ and $\mathrm{R}(t)$ but leads to a counterintuitive decrease in overall $\mathrm{Acc}(t)$.

Rule representation has a mild effect. In the airline and tax domains, some rules are represented as Markdown tables. To test whether representation format affects performance, we convert these tabular rules into textual "if-then" statements and compare with original results. The comparison shows that converting tabular rules into text slightly improves $\mathrm{R}(r)$, but has little impact on other metrics, including $\mathrm{Acc}(t)$.

Distractive rules significantly impairs LLM performance. An essential aspect of rule-following involves identifying which rules are relevant to the current problem. We assess the extent to which irrelevant rules detrimentally affect performance, and notice that the presence of distractive (irrelevant) rules significantly degrades LLM performance.

External tools for math and logic helps in rule application. The most simple way to reduce rule application errors is to introduce external tools for mathematical and logical calculations. We examine the benefit of tool augmentation by using the Python interpreter as an oracle tool for math and logic and prompting LLMs to output Python codes as their reasoning process, and observe a significant performance boost despite of non-perfect performance still.

4.3.2 Case Studies

We further conduct a failure case study (in Appendix D.3) to provide an intuitive understanding of how and why LLMs fail in complex rule-following problems. **We summarize three common failure modes of LLMs:**

(1) LLMs may fail to recall certain rules.

(2) LLMs can get confused by similar rules.

(3) LLMs sometimes compute incorrect results.

5 Conclusions

In this paper, we introduce RuleArena, a real-world benchmark to assess the abilities of LLMs on various rule-guided reasoning tasks. We observe that existing LLMs face significant challenges when they tackle problems on RuleArena - even the strongest Claude-3.5 and GPT-4o models can hardly succeed on our hardest tasks. Our further analysis indicates that LLMs struggle to integrate multiple rules or facts cohesively, are prone to irrelevant distractions, and can benefit significantly from external tools. RuleArena poses fundamental challenges in complex rule following, and provides a valuable tool for understanding and enhancing LLM reasoning. We envision RuleArena as a foundation for future works to improve LLM capacities in solving increasingly complex tasks.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 2357–2367, 2019. URL https://arxiv.org/abs/1905.13319.

Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 2024. URL https://docs.anthropic.com/en/docs/resources/model-card.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2211.12588.

Yihan Chen, Benfeng Xu, Quan Wang, Yi Liu, and Zhendong Mao. Benchmarking large language models on controllable generation under diversified instructions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17808–17816, 2024a. URL https://arxiv.org/abs/2401.00690.

Yihan Chen, Benfeng Xu, Quan Wang, Yi Liu, and Zhendong Mao. Benchmarking large language models on controllable generation under diversified instructions. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 17808–17816, 2024b. URL https://arxiv.org/abs/2401.00690.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022. URL https://arxiv.org/abs/2301.00234.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya

Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. Nphardeval: Dynamic benchmark on reasoning ability of large language models via complexity classes. *arXiv preprint arXiv:2312.14890*, 2023. URL https://arxiv.org/abs/2312.14890.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. Neural module networks for reasoning over text. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020. URL https://arxiv.org/abs/1912.04971.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq R. Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir Radev. FOLIO: natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022. URL https://arxiv.org/abs/2209.00840.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18188–18196, 2024. URL https://arxiv.org/abs/2309.09150.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS: Datasets and Benchmarks)*, 2021. URL https://arxiv.org/abs/2103.03874.

Wenyue Hua, Kaijie Zhu, Lingyao Li, Lizhou Fan, Shuhang Lin, Mingyu Jin, Haochen Xue, Zelong Li, JinDong Wang, and Yongfeng Zhang. Disentangling logic: The role of context in large language model reasoning capabilities. *arXiv preprint arXiv:2406.02787*, 2024. URL https://arxiv.org/abs/2406.02787.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4667–4688, 2024. URL https://arxiv.org/abs/2310.20410.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2205.11916.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 1152–1157, 2016. URL https://arxiv.org/abs/2311.04235.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models, 2023. URL https://github.com/tatsu-lab/alpaca_eval.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 158–167, 2017. URL https://arxiv.org/abs/1705.04146.

Xiao Liu, Xuanyu Lei, Shengyuan Wang, Yue Huang, Zhuoer Feng, Bosi Wen, Jiale Cheng, Pei Ke, Yifan Xu, Weng Lam Tam, et al. Alignbench: Benchmarking chinese alignment of large language models. *arXiv preprint arXiv:2311.18743*, 2023. URL https://arxiv.org/abs/2311.18743.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019. URL `https://arxiv.org/abs/1904.12584`.

Norman Mu, Sarah Chen, Zifan Wang, Sizhe Chen, David Karamardian, Lulwa Aljeraisy, Dan Hendrycks, and David A. Wagner. Can llms follow simple rules? *arXiv preprint arXiv:2311.04235*, 2023. URL `https://arxiv.org/abs/2311.04235`.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL `https://arxiv.org/abs/2303.08774`.

OpenAI. Hello gpt-4o. *OpenAI Blogs*, 2024. URL `https://openai.com/index/hello-gpt-4o`.

Qwen Team. Qwen2.5: A party of foundation models, 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Frederick Wieting, Nanyun Peng, and Xuezhe Ma. Evaluating large language models on controlled generation tasks. *arXiv preprint arXiv:2310.14542*, 2023. URL `https://arxiv.org/abs/2310.14542`.

Wangtao Sun, Chenxiang Zhang, Xueyou Zhang, Ziyang Huang, Haotian Xu, Pei Chen, Shizhu He, Jun Zhao, and Kang Liu. Beyond instruction following: Evaluating rule following of large language models. *arXiv preprint arXiv:2407.08440*, 2024. URL `https://arxiv.org/abs/2407.08440`.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021*, pp. 3621–3634, 2021. URL `https://arxiv.org/abs/2012.13048`.

Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. Struc-bench: Are large language models really good at generating complex structured data? *arXiv preprint arXiv:2309.08963*, 2023. URL `https://arxiv.org/abs/2309.08963`.

Google Gemini Team. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. URL `https://arxiv.org/abs/2312.11805`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. URL `https://arxiv.org/abs/2302.13971`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL `https://arxiv.org/abs/2201.11903`.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023. URL `https://arxiv.org/abs/2303.03846`.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. Benchmarking complex instruction-following with multiple constraints composition. *arXiv preprint arXiv:2407.03978*, 2024. URL `https://arxiv.org/abs/2407.03978`.

Xiaobao Wu, Liangming Pan, William Yang Wang, and Anh Tuan Luu. AKEW: Assessing knowledge editing in the wild. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15118–15133, 2024a. URL `https://aclanthology.org/2024.emnlp-main.843`.

Xiaobao Wu, Liangming Pan, Yuxi Xie, Ruiwen Zhou, Shuai Zhao, Yubo Ma, Mingzhe Du, Rui Mao, Anh Tuan Luu, and William Yang Wang. Antileak-bench: Preventing data contamination by automatically constructing benchmarks with updated real-world knowledge. *arXiv preprint arXiv:2412.13670*, 2024b.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. Fofo: A benchmark to evaluate llms' format-following capability. *arXiv preprint arXiv:2402.18667*, 2024. URL https://arxiv.org/abs/2402.18667.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023. URL https://arxiv.org/abs/2304.12244.

Shaokun Zhang, Xiaobo Xia, Zhaoqing Wang, Ling-Hao Chen, Jiale Liu, Qingyun Wu, and Tongliang Liu. Ideal: Influence-driven selective annotations empower in-context learners in large language models. *arXiv preprint arXiv:2310.10873*, 2023. URL https://arxiv.org/abs/2310.10873.

Xiang Zhang and Dujian Ding. Supervised chain of thought. *arXiv preprint arXiv:2410.14198*, 2024. URL https://arxiv.org/abs/2410.14198.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2306.05685.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Yining Chen, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. Analytical reasoning of text. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 2306–2319, 2022. URL https://arxiv.org/abs/2104.06598.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. Dyval: Graph-informed dynamic evaluation of large language models. *arXiv preprint arXiv:2309.17167*, 2023. URL https://arxiv.org/abs/2309.17167.

A Terminology Explanation in NBA

We briefly explain the NBA terminologies mentioned in this paper as follows:

Salary Cap. The Salary Cap of NBA is a rule that limits how much money each team can spend on player salaries. It is designed to keep teams on a level playing field financially, so wealthier teams cannot just purchase all the best players. The league sets the cap based on its overall revenue.

(Salary Cap) Exceptions. The NBA uses a "soft" Salary Cap, meaning teams can exceed the limit using certain Exceptions. Following are some commonly used Exceptions:

- **Mid-Level Exception (MLE)** allows teams to sign free players even if they are above the salary cap. There are three types of MLEs, i.e. Non-Taxpayer MLE, Taxpayer MLE, and MLE for Room Teams, applicable to teams in different salary situations.
- **Traded Player Exceptions (TPE)** is a tool that allows teams to make trades even if they are over the salary cap. When a team trades a player for less salary than it gives away (or for nothing), it creates a TPE, which is like a "credit" they can use later. If a team wants to acquire more salaries than it gives away in a trade, it can also use certain types of TPE to make such trade.
- **Veteran Free Agent Exception (Bird Rights)** in the NBA allow teams to re-sign their own players even if they are over the salary cap. Named after Larry Bird, this rule encourages teams to keep their star players. There are three types of Bird Rights, i.e. Bird Rights, Early Bird Rights, and Non-Bird Rights, applicable to players that play for the same team for different numbers of consecutive seasons.

B Data Collection and Annotation

B.1 Rule Collection

Airline. We collect the policy for bag and optional fees from *American Airlines*[8]. Specifically, the rules in the policy mainly include: 1) the allowance of carry-on luggage; 2) the base price for checking each luggage on different routes and in different cabin classes; 3) the additional fees for luggage overweight or oversize to varying degrees on different routes and in different cabin classes; 4) when calculating fees for overweight and oversize luggage for each piece, only the higher of the two should apply. Many rules (base price, overweight/oversize fees) in this domain are represented in tabular forms, and we regard one entire table as one rule.

NBA. We collect the regulations for NBA transactions from *2023 NBA Collective Bargaining Agreements*[9] (CBA) and excerpt from the *NBA Constitution and By-Laws*[10] regarding the rules for trading first-round draft picks (i.e., the Stepien Rule). Since the complete CBA is too long (676 Pages PDF), we only aggregate the most commonly used rules such as the limits on salary and length of player contract, on team salary, and on player contract trade among teams. As applying rules of the same type but applicable under different conditions may result in completely different subsequent reasoning process, different from in airline domain, we depart such one paragraph including such similar rules into separate rules.

Tax. We collect tax forms and relevant instructions from *Internal Revenue Service* (IRS)[11]. Starting from the most famous Form 1040 (U.S. Individual Income Tax Return) and its basic Schedules 1-3, we consider more complex settings commonly happen in real-life, including using itemized deductions (Schedule A), self-employment (Schedule C and Schedule SE), education expenses and/or credits (Form 8863), and child and/or other dependent credits (Schedule 8812). We treat each line in these forms and its instructions as one rule, and convert the forms into line numbers and text for each line as LLM input.

[8] https://www.aa.com/i18n/customer-service/support/optional-service-fees.jsp

[9] https://ak-static.cms.nba.com/wp-content/uploads/sites/4/2023/06/2023-NBA-Collective-Bargaining-Agreement.pdf

[10] https://ak-static-int.nba.com/wp-content/uploads/sites/3/2015/12/NBA-Constitution-and-By-Laws.pdf

[11] https://www.irs.gov/forms-instructions

B.2 NBA Data Annotation

For NBA tasks, we first survey famous rules and transactions that have happened in the NBA in recent 30 years and decide the 54 rules used in our annotation. To balance the task difficulty and annotation difficulty, we further simplify the rules by unifying different types of team salary (defined in different rules and calculated in different ways) into one simple "Team Salary". The process of annotating one problem is described as follows:

Creating team and player situations. Our annotators are first required to create diverse valid scenarios involving one or more teams and players, as the following "team_situations" and "players_situations", and provide the number of teams ("n_teams") and players ("n_players") involved. Each item in the "team_situations" list indicates the current salary of the team and its available first-round draft picks, while each item in the"player_situations" list tells the player's information including his draft year, age, and current (or last) contract.

Writting transactions. Next, our annotators write "n_operations" sentences in the "operations" list, where each item corresponding to one team signing a player or several teams conducting a trade, and determine whether all these transactions are allowed according to the rules. The "answer" should be **True** if all transactions are allowed otherwise **False**. If "answer" is **False**, we ask our annotators to further provide "illegal_team" and "illegal_operation" as the specific team and transaction component that violates the rules.

Listing relevant rules. Finally, our annotators are told to provide a list of "relevant_rules" including all rules that they believe should be involved if humans need to consider the case comprehensively.

```
{
    "n_teams": int = ...,
    "n_players": int = ...,
    "n_operations": int = ...,
    "team_situations": list[str] = [...],
    "player_situations": list[str] = [...],
    "operations": list[str] = [...],
    "answer": bool = ...,
    "illegal_operation": str = ...,
    "illegal_team": str = ...,
    "relevant_rules": list[str] = [...]
}
```

The format of annotated NBA test problems.

C Structured Rule Extraction in Each Scenario

As introduced in Section 4.1, we utilize the structured output mode of GPT-4o (OpenAI, 2024) to convert LLMs' textual output into structured data. Here we present the data structure we used in parsing.

Airline. In airline domain we ask LLMs to parse the the list of checked luggage as well as provided basic information.

```
class BagCost(BaseModel):
    size: int
    weight: int
    base_check_fee: int
    oversize_fee: int
    overweight_fee: int
    total_fee: int

class PassengerClass(str, Enum):
    be = "Basic Economy"
    main = "Main Cabin"
    mp = "Main Plus"
    pe = "Premium Economy"
    business = "Business"
    first = "First"

class Response(BaseModel):
    passenger_class: str
    place_of_departure: str
    place_of_arrival: str
    ticket_price: int
    checked_bags: list[BagCost]
    total_cost: int
```

NBA. In NBA domain we let the LLM parser decide whether each of the 54 rules is applied.

```python
class RuleExtraction(BaseModel):
    # contract length
    contract_length_at_most_4_year_except_qualifying_veteran_free_agent_5_year: bool
    contract_length_at_most_2_year_bi_annual_exception: bool
    contract_length_at_most_4_year_non_taxpayer_mid_level_exception: bool
    contract_length_at_most_2_year_taxpayer_mid_level_exception: bool
    contract_length_at_most_3_year_mid_level_exception_for_room_team: bool
    contract_length_at_most_2_year_minimum_player_salary_exception: bool

    # basic rules
    salary_cap_no_exceed_without_exception: bool
    maximum_salary_for_player_less_than_7_year_service: bool
    maximum_salary_for_player_7_to_9_year_service: bool
    maximum_salary_for_player_10_or_more_year_service: bool
    higher_max_criterion_for_5th_year_eligible_player: bool
    salary_increase_and_decrease_ratio_except_qualiyfing_or_early_qualifying_veteran_free_agent: bool
    salary_increase_and_decrease_ratio_for_qualiyfing_or_early_qualifying_veteran_free_agent: bool

    # 38 year old provision
    defer_compensation_38_year_old: bool
    defer_compensation_qualifying_veteran_free_agent_38_year_old: bool

    # apron level as hard cap rules
    bi_annual_exception_hard_cap_first_apron_level: bool
    non_taxpayer_mid_level_exception_hard_cap_first_apron_level: bool
    sign_and_trade_hard_cap_first_apron_level: bool
    expanded_traded_player_exception_hard_cap_first_apron_level: bool
    aggregated_traded_player_exception_hard_cap_second_apron_level: bool
    cash_in_trade_hard_cap_second_apron_level: bool
    sign_and_trade_assigner_traded_player_exception_hard_cap_second_apron_level: bool
    taxpayer_mid_level_exception_hard_cap_second_apron_level: bool
    traded_player_exception_250k_reduced_first_apron_level: bool

    # exceptions
    # bird rights
    qualifying_veteran_free_agent_exception: bool
    early_qualifying_veteran_free_agent_exception: bool
    non_qualifying_veteran_free_agent_exception: bool
    salary_space_consumption_qualifying_veteran_free_agent: bool
    salary_space_consumption_early_qualifying_veteran_free_agent: bool
    salary_space_consumption_non_qualifying_veteran_free_agent: bool
    salary_space_consumption_standard_traded_player_exception: bool

    # bi-annual exception
    bi_annual_exception: bool

    # mid level exceptions
    non_taxpayer_mid_level_exception: bool
    taxpayer_mid_level_exception: bool
    mid_level_exception_for_room_team: bool
    minimum_player_salary_exception: bool

    # traded player exceptions
    standard_traded_player_exception: bool
    aggregated_standard_traded_player_exception: bool
    expanded_traded_player_exception: bool
    traded_player_exception_for_room_team: bool
    traded_player_exception_only_one_minimum_traded_player_under_conditions: bool

    # trade rules
    pay_or_receive_cash_maximum_in_a_year: bool
    rookie_or_two_way_contract_cannot_be_traded_within_30_days: bool
    free_agent_sign_contract_cannot_be_traded_within_3_month_or_before_dec_15: bool
    qualifying_or_early_qualifying_free_agent_sign_contract_cannot_be_traded_within_3_month_or_before_jan_15:
bool

    # sign-and-trade rules
    sign_and_trade_3_to_4_year: bool
    sign_and_trade_not_with_mid_level_exception: bool
    sign_and_trade_no_higher_than_25_percent_for_higher_max_5th_year_eligible_player: bool
    sign_and_trade_assignee_team_has_room: bool
    sign_and_trade_qualifying_free_agent_half_salary_for_traded_player_exception: bool

    # restricted free agent rules (Arenas provision)
    offer_sheet_for_1_or_2_year_service_player_no_more_than_mid_level_in_first_2_year: bool
    offer_sheet_for_1_or_2_year_service_player_3rd_year_maximum_if_first_2_year_maximum: bool
    offer_sheet_for_1_or_2_year_service_player_4th_year_maximum_if_3_year: bool
    offer_sheet_for_1_or_2_year_service_player_average_salary_more_than_2_year: bool

    # first-round draft pick trade rules
    stepien_rule_no_sell_or_no_consecutive_first_round_draft_pick_trade: bool
```

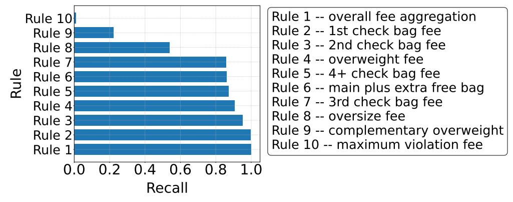
Tax. In tax domain we just list each line in Form 1040 and its Schedules 1-3 for parsing.

```python
class Form1040(BaseModel):
    name: str = Field(description="Name of taxpayer")
    age: int = Field(description="Age of taxpayer")
    spouse_age: int = Field(description="Age of taxpayer's spouse")
    filing_status: FilingStatus = Field(description="Filing status of taxpayer")
    blind: bool = Field(description="Taxpayer is blind")
    spouse_blind: bool = Field(description="Taxpayer's spouse is blind")
    itemized: bool = Field(description="Taxpayer uses itemized deductions")
    num_qualifying_children: int = Field(description="Number of qualifying children")
    num_other_dependents: int = Field(description="Number of other dependents")
    wage_tip_compensation: float = Field(description="Form 1040 Line 1a")
    household_employee_wage: float = Field(description="Form 1040 Line 1b")
    unreported_tip: float = Field(description="Form 1040 Line 1c")
    nontaxable_combat_pay: float = Field(description="Form 1040 Line 1d")
    wage_tip_compensation_total: float = Field(description="Form 1040 Line 1z")
    tax_exempt_interest: float = Field(description="Form 1040 Line 2a")
    taxable_interest: float = Field(description="Form 1040 Line 2b")
    qualified_dividends: float = Field(description="Form 1040 Line 3a")
    ordinary_dividends: float = Field(description="Form 1040 Line 3b")
    ira_distributions: float = Field(description="Form 1040 Line 4a")
    taxable_ira_distributions: float = Field(description="Form 1040 Line 4b")
    all_pensions: float = Field(description="Form 1040 Line 5a")
    taxable_pensions: float = Field(description="Form 1040 Line 5b")
    social_security_benefits: float = Field(description="Form 1040 Line 6a")
    taxable_social_security_benefits: float = Field(description="Form 1040 Line 6b")
    capital_gain_or_loss: float = Field(description="Form 1040 Line 7")
    additional_income: float = Field(description="Form 1040 Line 8")
    total_income: float = Field(description="Form 1040 Line 9")
    total_adjustments: float = Field(description="Form 1040 Line 10")
    adjusted_gross_income: float = Field(description="Form 1040 Line 11")
    standard_or_itemized_deductions: float = Field(description="Form 1040 Line 12")
    qualified_business_income: float = Field(description="Form 1040 Line 13")
    total_deductions: float = Field(description="Form 1040 Line 14")
    computed_taxable_income: float = Field(description="Form 1040 Line 15")
    taxes: float = Field(description="Form 1040 Line 16")
    copy_schedule_2_line_3: float = Field(description="Form 1040 Line 17")
    f1040_line_18: float = Field(description="Form 1040 Line 18")
    ctc_or_other_dependent_credit: float = Field(description="Form 1040 Line 19")
    copy_schedule_3_line_8: float = Field(description="Form 1040 Line 20")
    accumulated_credits: float = Field(description="Form 1040 Line 21")
    taxes_after_credits: float = Field(description="Form 1040 Line 22")
    other_taxes: float = Field(description="Form 1040 Line 23")
    total_tax: float = Field(description="Form 1040 Line 24")
    federal_income_tax_withheld: float = Field(description="Form 1040 Line 25")
    earned_income_credit: float = Field(description="Form 1040 Line 27")
    additional_child_tax_credit: float = Field(description="Form 1040 Line 28")
    american_opportunity_credit: float = Field(description="Form 1040 Line 29")
    copy_schedule_3_line_15: float = Field(description="Form 1040 Line 31")
    total_other_payments_and_refundable_credits: float = Field(description="Form 1040 Line 32")
    total_payments: float = Field(description="Form 1040 Line 33")
    amount_owed_or_overpaid: float = Field(description="Form 1040 Line 37 (negative if overpaid)")
    taxable_state_refunds: float = Field(description="Schedule 1 Line 1")
    alimony_income: float = Field(description="Schedule 1 Line 2a")
    sale_of_business: float = Field(description="Schedule 1 Line 4")
    rental_real_estate_sch1: float = Field(description="Schedule 1 Line 5")
    farm_income: float = Field(description="Schedule 1 Line 6")
    unemployment_compensation: float = Field(description="Schedule 1 Line 7")
    other_income: float = Field(description="Schedule 1 Line 8")
    educator_expenses: float = Field(description="Schedule 1 Line 11")
    hsa_deduction: float = Field(description="Schedule 1 Line 13")
    self_employment_deductible: float = Field(description="Schedule 1 Line 15")
    ira_deduction: float = Field(description="Schedule 1 Line 20")
    student_loan_interest_deduction: float = Field(description="Schedule 1 Line 21")
    other_adjustments: float = Field(description="Schedule 1 Line 24")
    amt_f6251: float = Field(description="Schedule 2 Line 1")
    credit_repayment: float = Field(description="Schedule 2 Line 2")
    schedule_2_total_taxes: float = Field(description="Schedule 2 Line 3 (= Line 1 + Line 2)")
    self_employment_tax: float = Field(description="Schedule 2 Line 4")
    other_additional_taxes: float = Field(description="Schedule 2 Line 17")
    schedule_2_total_other_taxes: float = Field(description="Schedule 2 Line 21 (= Line 4 + Line 17)")
    foreign_tax_credit: float = Field(description="Schedule 3 Line 1")
    dependent_care: float = Field(description="Schedule 3 Line 2")
    computed_education_credits: float = Field(description="Schedule 3 Line 3")
    retirement_savings: float = Field(description="Schedule 3 Line 4")
    elderly_disabled_credits: float = Field(description="Schedule 3 Line 6d")
    plug_in_motor_vehicle: float = Field(description="Schedule 3 Line 6i")
    alt_motor_vehicle: float = Field(description="Schedule 3 Line 6j")
    schedule_3_line_8: float = Field(description="Schedule 3 Line 8")
    medical_dental_expenses: Optional[float] = Field(description="Schedule A Line 1 (if itemized)")
    state_local_income_or_sales_tax: Optional[float] = Field(description="Schedule A Line 5a (if itemized)")
    state_local_real_estate_tax: Optional[float] = Field(description="Schedule A Line 5b (if itemized)")
    state_local_personal_property_tax: Optional[float] = Field(description="Schedule A Line 5c (if itemized)")
    other_taxes_paid: Optional[float] = Field(description="Schedule A Line 6 (if itemized)")
    home_mortgage_interest_and_points: Optional[float] = Field(description="Schedule A Line 8a (if itemized)")
    home_mortgage_interest_unreported: Optional[float] = Field(description="Schedule A Line 8b")
    home_mortgage_points_unreported: Optional[float] = Field(description="Schedule A Line 8c (if itemized)")
    investment_interest: Optional[float] = Field(description="Schedule A Line 9 (if itemized)")
    charity_cash: Optional[float] = Field(description="Schedule A Line 11 (if itemized)")
    charity_non_cash: Optional[float] = Field(description="Schedule A Line 12 (if itemized)")
    casualty_and_theft_loss: Optional[float] = Field(description="Schedule A Line 15 (if itemized)")
    other_itemized_deductions: Optional[float] = Field(description="Schedule A Line 16 (if itemized)")
    gross_receipts: Optional[float] = Field(description="Schedule C Line 1 (if self-employed)")
    returns_and_allowances: Optional[float] = Field(description="Schedule C Line 2 (if self-employed)")
    cost_of_goods_sold: Optional[float] = Field(description="Schedule C Line 4 (if self-employed)")
    other_inc_sched_c: Optional[float] = Field(description="Schedule C Line 6 (if self-employed)")
    total_expenses: Optional[float] = Field(description="Schedule C Line 28 (if self-employed)")
    expenses_of_home: Optional[float] = Field(description="Schedule C Line 30 (if self-employed)")
    net_profit: Optional[float] = Field(description="Schedule C Line 31 (if self-employed)")
    total_social_security_wages: Optional[float] = Field(description="Schedule SE Line 8 (if self-employed)")
    student_list: Optional[list[Student]] = Field(description="List of students with education expenses")
```

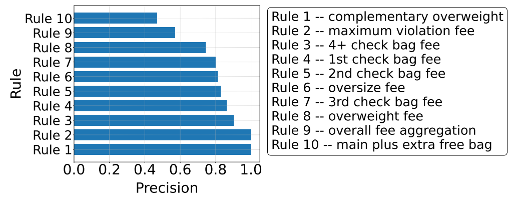
D More Experiment Results and Analysis

D.1 Rule-Wise Statistics

We visualize the rule-wise *recall*, *precision*, and *correctness* in Figure 3-5. Since *precision* is always 1.0 in airline and tax domains, we skip these two charts.

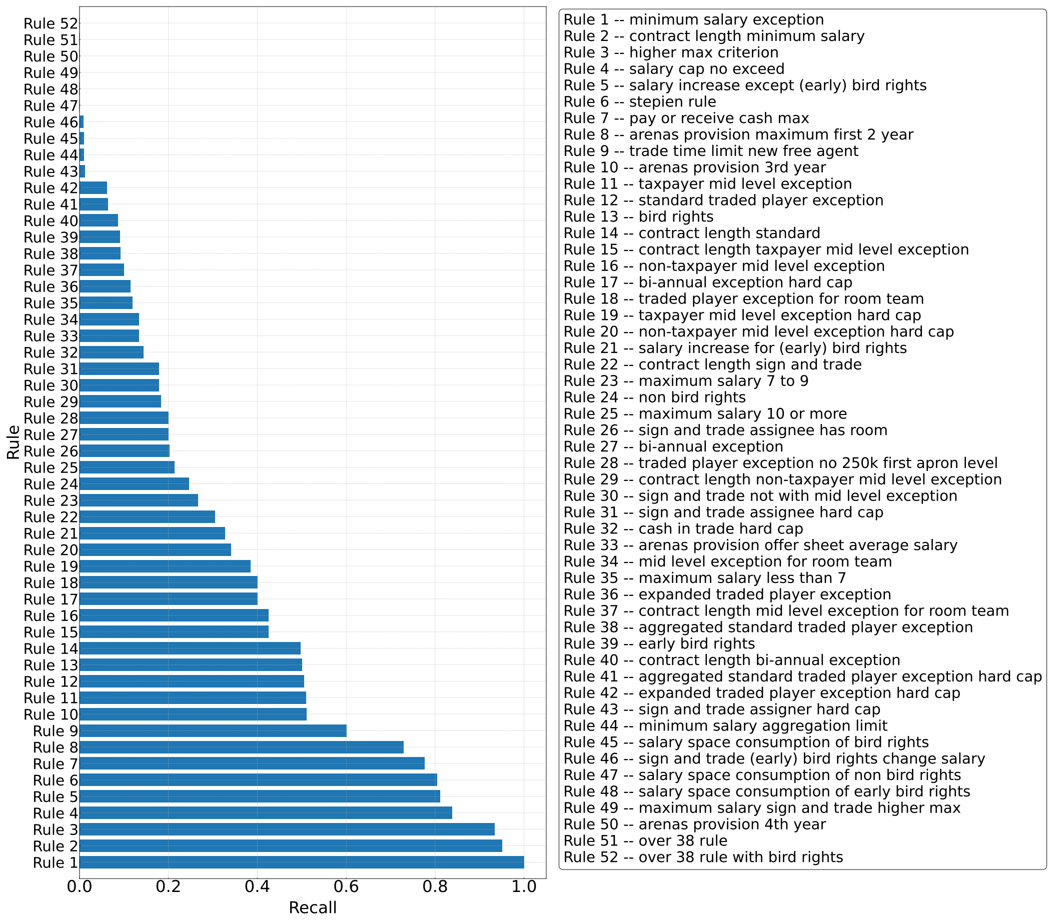


(a) Recall

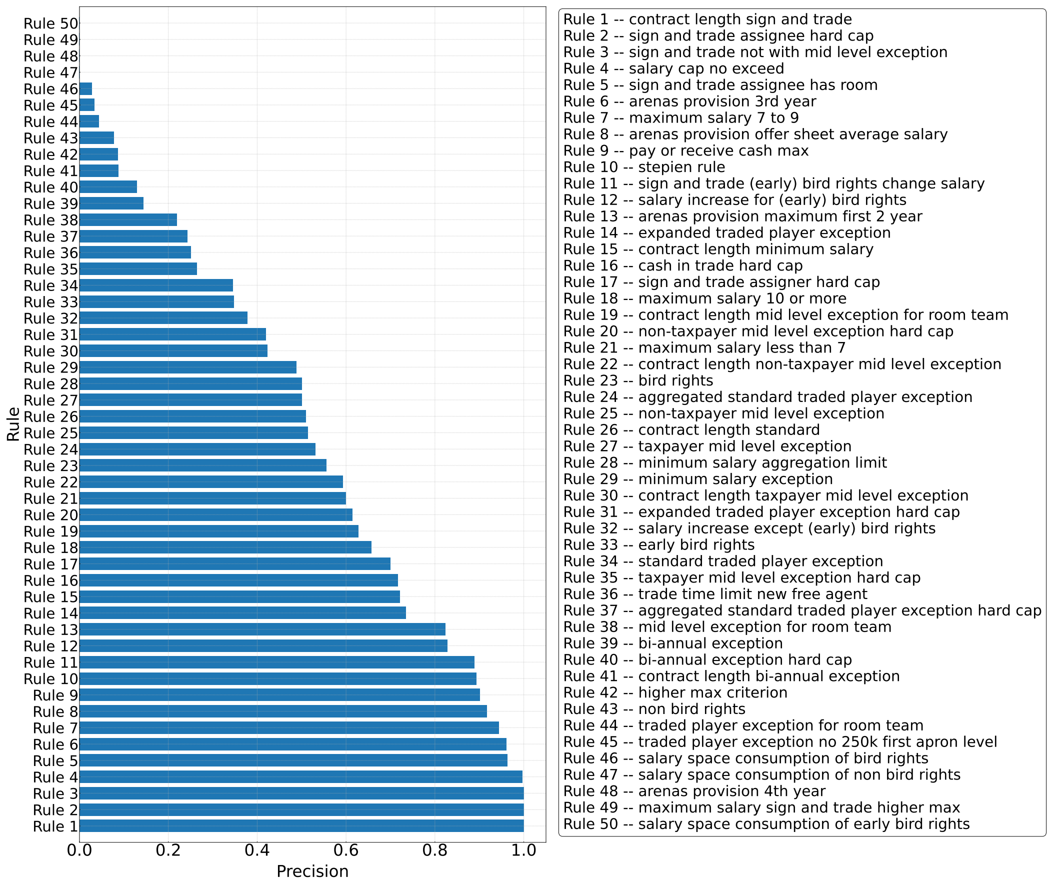


(b) Correctness

Figure 3: Rule-wise metrics of rules in airline domain.

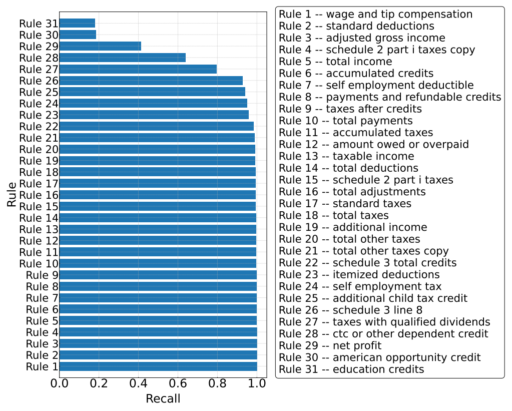


(a) Recall

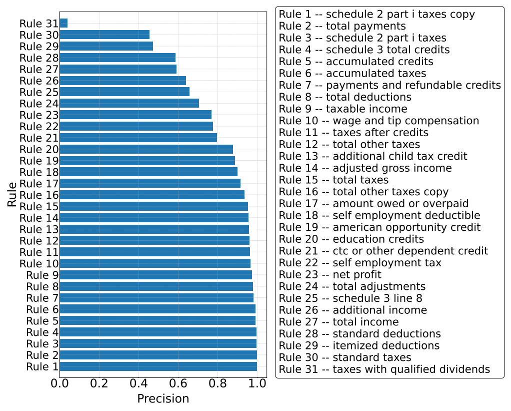


(b) Precision

Figure 4: Rule-wise metrics of rules in NBA domain.



(a) Recall



(b) Correctness

Figure 5: Rule-wise metrics of rules in tax domain.

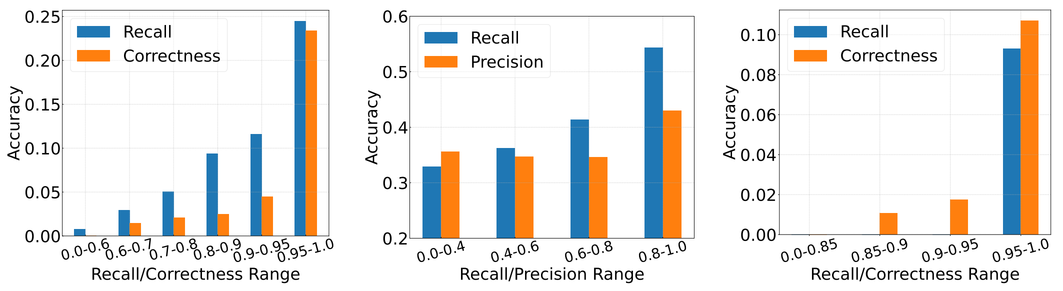
D.2 WHAT IMPACTS RULE FOLLOWING?

In this section, we investigate the factors influencing LLM performance, as measured by $\mathrm{Acc}(t)$. We begin by examining the correlation between $\mathrm{Acc}(t)$ and other key metrics, including $\mathrm{P}(t)$, $\mathrm{AC}(t)$, and $\mathrm{R}(t)$. We then consider the effects of in-context examples, different rule representations, and the presence of distractors.

D.2.1 CORRELATION BETWEEN ACCURACY AND OTHER METRICS

To understand which factors most directly affect $\mathrm{Acc}(t)$, we visualize its correlation with other metrics in Figure 6 across all three domains on datapoints from all difficulty levels. From Figure 6a and Figure 6b, we observe an almost linear relationship between $\mathrm{R}(t)$ and $\mathrm{Acc}(t)$. Notice that in the tax domain (Figure 6c), a recall lower than 0.95 immediately results in zero accuracy.

In contrast, the correlation between $\mathrm{AC}(t)$ and $\mathrm{Acc}(t)$ is highly non-linear, as seen in Figure 6a and Figure 6c. In many cases, a single computational error in rule application (thus reducing $\mathrm{AC}(t)$) is sufficient to produce an incorrect final answer, indicating that only near-perfect $\mathrm{AC}(t)$ leads to significant $\mathrm{Acc}(t)$ improvements. For the NBA domain, we also compare $\mathrm{P}(t)$ and $\mathrm{Acc}(t)$; since $\mathrm{P}(t)$ is always 100% for the airline and tax domains, these correlations are not meaningful there. We find no clear relationship between $\mathrm{P}(t)$ and $\mathrm{Acc}(t)$ for the NBA problems (Figure 6b).



(a) Airline (b) NBA (c) Tax

Figure 6: Correlation between problem-wise metrics and accuracy. The correlation is the most obvious and almost linear between $\mathrm{R}(t)$ and $\mathrm{Acc}(t)$, while highly non-linear or unclear between other two metrics and $\mathrm{Acc}(t)$.

D.2.2 DO IN-CONTEXT EXAMPLES HELP?

Table 2 presents the results with or without a level-1 1-shot example. LLMs generally provide better performances given 1-shot example on airline, tax, and (easy) NBA problems. Many studies have shown the benefit of in-context learning (Dong et al., 2022; Wei et al., 2023; Zhang et al., 2023), which conforms with our observation that $\mathrm{Acc}(t)$ gets higher in the 1-shot setting. This performance boost comes from both the enhancement of $\mathrm{AC}(t)$ as well as a better understanding of the reasoning process, indicated by higher $\mathrm{R}(t)$.

However, when tackling more challenging NBA problems (Levels 2 and 3), providing an example increases $\mathrm{P}(t)$ and $\mathrm{R}(t)$ but leads to a counterintuitive decrease in overall $\mathrm{Acc}(t)$. This improvement in precision and recall primarily arises from the non-essential rules included in the in-context example, such as the "Over 38 rule" and "Salary consumption of veteran free agent". We compute the $\mathrm{R}(r)$ and $\mathrm{P}(r)$ for these two rules as in Table 7.

Notably, while the $\mathrm{R}(r)$ for both rules improves, $\mathrm{P}(r)$ for rule "Salary consumption" is much lower. This shows that thought the in-context example does remind LLMs to apply rules that they might overlook, some rules like "Salary consumption" can be too hard for LLMs to understand even taught by an expert example, and thus LLMs do not understand what scenarios are suitable for such rules to apply. In addition, we find the performance on the remaining rules remains mostly unchanged. The exact cause of the performance decline in accuracy is difficult to pinpoint as our annotation on NBA does not contain detailed intermediate reasoning annotations. However, prior work (Fan

Rule	**Setting**	$\mathrm{R}(r)$	$\mathrm{P}(r)$
Over 38 rule	0-shot	0.00	N/A
	1-shot	0.35	0.76
Salary consumption	0-shot	0.00	N/A
	1-shot	0.23	0.20

Table 7: 0-shot and 1-shot rule-wise comparison.

et al., 2023) suggests that if the in-context example is "easier" than the target problem, the example can inadvertently degrade performance—a plausible explanation for why accuracy drops even as precision and recall improve.

D.2.3 Does Rule Representation Matter?

In the airline and tax domains, some rules are represented as Markdown tables. To test whether representation format affects performance, we convert these tabular rules into textual "if-then" statements. Table 8 shows that converting tabular rules into text improves $\mathrm{R}(r)$, but has little impact on other metrics, including $\mathrm{Acc}(t)$.

Models	**Setting**	**Airline**			**Tax**		
		$\mathrm{AC}(t)$	$\mathrm{R}(t)$	$\mathrm{Acc}(t)$	$\mathrm{AC}(t)$	$\mathrm{R}(t)$	$\mathrm{Acc}(t)$
Llama 70B	Table	0.764	0.558	0.01	0.834	0.989	0.01
	Text	0.764	0.582	0.01	0.814	0.991	0.00
Qwen 72B	Table	0.636	0.586	0.01	0.888	0.998	0.10
	Text	0.748	0.633	0.02	0.859	0.996	0.01
Llama 405B	Table	0.854	0.604	0.03	0.923	0.999	0.16
	Text	0.835	0.587	0.07	0.919	0.998	0.05
Claude-3.5	Table	0.930	0.702	0.04	0.964	1.000	0.32
	Text	0.937	0.705	0.06	0.971	1.000	0.33
GPT-4o	Table	0.862	0.616	0.02	0.965	1.000	0.42
	Text	0.864	0.669	0.03	0.960	1.000	0.33

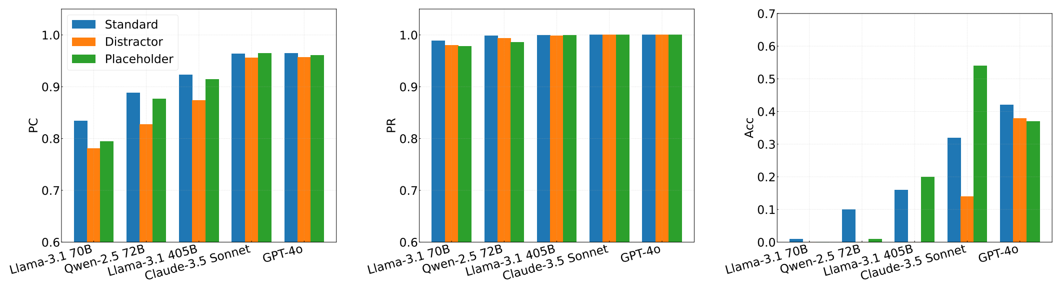
Table 8: Results of different LLMs given different rule representations.

D.2.4 Do Distractive Rules Matter?

An essential aspect of rule-following involves identifying which rules are relevant to the current problem. In our experiments, all domain-specific rules are provided in the prompt, leaving it to the LLMs to determine which ones should be applied. To assess the extent to which irrelevant rules detrimentally affect performance, we focus on the tax scenario. In this domain, we can introduce additional tax forms that contain only zero values, effectively rendering any corresponding rules irrelevant. Despite these rules being unnecessary, their mere presence may mislead LLMs into treating them as important.

To isolate the effect of these distractive rules from the influence of increased context length, we also create a "Placeholder" setting. In this setting, we replace the distractive rules with an equivalent amount of meaningless tokens that do not correspond to any rules. By comparing performance under these two conditions, we can distinguish between the impact of irrelevant rules and the general challenge posed by a longer input.

As shown in Figure 7, the presence of distractive (irrelevant) rules significantly degrades LLM performance, while increasing context length using meaningless placeholders results in only a minor performance drop. These findings suggest that LLMs remain vulnerable to distraction, which undermines their reliability when confronted with superfluous, yet superficially valid, rules.



(a) Correctness (b) Recall (c) Accuracy

Figure 7: The effect of distractive rules and context length. The "Standard" mode refers to the default setting of Level 1 tax problems, the "Distractor" mode appends nullified forms after the "Standard" input, and the "Placeholder" mode adds meaningless tokens on space lines. Distractive rules lead to a significant drop on the performances of all LLMs, while meaningless tokens make little difference to the performance.

D.2.5 Can Tool Augmentation Help?

In Appendix D.2.1, we notice that only near-perfect application correctness $\mathrm{AC}(t)$ can lead to significant accuracy improvement. As the most simple way to reduce errors, especially in mathematical and logical operations, is to introduce external tools, we wonder to what extent external tools can help in our rule-guided reasoning tasks.

Following program of thoughts (Chen et al., 2023) prompting, we ask our LLMs to write Python code to calculate the answer on airline bag fee tasks by defining a `solution()` function and returning the `total_cost` variable, and the execution result of `solution()` function is viewed as the predicted answer. In this way, the Python interpreter can be viewed as an oracle tool for mathematical and logical calculation. To ensure the correct format of response, we use the 1-shot setting, so we compare the additional results with the original 1-shot results in Table 2 as follows:

Models	**Setting**	**Level 1**			**Level 2**		
		$\mathrm{AC}(t)$	$\mathrm{R}(t)$	$\mathrm{Acc}(t)$	$\mathrm{AC}(t)$	$\mathrm{R}(t)$	$\mathrm{Acc}(t)$
Llama 70B	1-shot Default	0.809	0.787	0.17	0.827	0.801	0.07
	Tool Augmented	**0.863**	**0.882**	**0.34**	0.827	**0.887**	**0.18**
Qwen 72B	1-shot Default	0.836	**0.908**	0.19	0.818	**0.901**	0.10
	Tool Augmented	**0.939**	0.899	**0.42**	**0.946**	0.899	**0.26**
GPT-4o	1-shot Default	0.922	0.885	0.32	0.875	0.853	0.16
	Tool Augmented	**0.939**	**0.914**	**0.44**	**0.937**	**0.940**	**0.33**

Table 9: Results of different LLMs with tool augmentation on airline tasks.

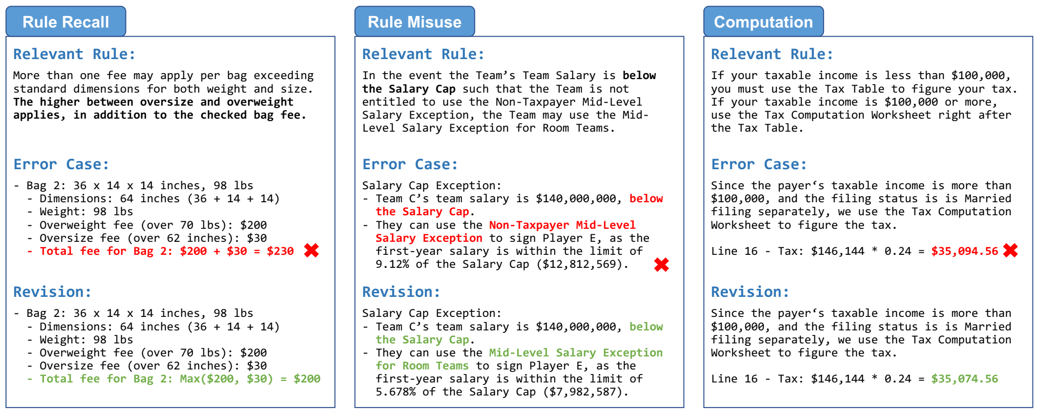
As can be seen from these results, when provided with oracle math and logic tools, LLMs can achieve a significant performance boost in terms of accuracy $\mathrm{Acc}(t)$. However, even provided with such tools, LLMs are far from being able to resolve our rule-guided reasoning tasks. We observe non-perfect recall $\mathrm{R}(t)$ and correctness $\mathrm{AC}(t)$, which indicates that LLMs still make mistakes in generated codes.

D.2.6 Intermediate Summary

In summary, various factors—such as rule complexity, the presence of distractive information, and the difficulty gap between in-context examples and target problems—can profoundly influence LLM performance. Even when LLMs succeed in simpler conditions, challenges like complex mathematical reasoning, large amounts of extraneous rules, and non-ideal in-context samples can severely limit their effectiveness on RuleArena problems.

D.3 Case Studies

To gain an intuitive understanding of how and why LLMs fail in complex rule-following problems, we present representative failure cases in Figure 8. These examples highlight three frequently observed failure modes:



Rule Recall

Relevant Rule:

More than one fee may apply per bag exceeding standard dimensions for both weight and size. **The higher between oversize and overweight applies, in addition to the checked bag fee.**

Error Case:

```
- Bag 2: 36 x 14 x 14 inches, 98 lbs
  - Dimensions: 64 inches (36 + 14 + 14)
  - Weight: 98 lbs
  - Overweight fee (over 70 lbs): $200
  - Oversize fee (over 62 inches): $30
  - Total fee for Bag 2: $200 + $30 = $230 ✖
```

Revision:

```
- Bag 2: 36 x 14 x 14 inches, 98 lbs
  - Dimensions: 64 inches (36 + 14 + 14)
  - Weight: 98 lbs
  - Overweight fee (over 70 lbs): $200
  - Oversize fee (over 62 inches): $30
  - Total fee for Bag 2: Max($200, $30) = $200
```

Rule Misuse

Relevant Rule:

In the event the Team’s Team Salary is **below the Salary Cap** such that the Team is not entitled to use the Non-Taxpayer Mid-Level Salary Exception, the Team may use the Mid-Level Salary Exception for Room Teams.

Error Case:

Salary Cap Exception:
- Team C’s team salary is $140,000,000, **below the Salary Cap**.
- They can use the **Non-Taxpayer Mid-Level Salary Exception** to sign Player E, as the first-year salary is within the limit of 9.12% of the Salary Cap ($12,812,569). ✖

Revision:

Salary Cap Exception:
- Team C’s team salary is $140,000,000, **below the Salary Cap**.
- They can use the **Mid-Level Salary Exception for Room Teams** to sign Player E, as the first-year salary is within the limit of 5.678% of the Salary Cap ($7,982,587).

Computation

Relevant Rule:

If your taxable income is less than $100,000, you must use the Tax Table to figure your tax. If your taxable income is $100,000 or more, use the Tax Computation Worksheet right after the Tax Table.

Error Case:

Since the payer‘s taxable income is more than $100,000, and the filing status is is Married filing separately, we use the Tax Computation Worksheet to figure the tax.

Line 16 - Tax: $146,144 * 0.24 = **$35,094.56** ✖

Revision:

Since the payer‘s taxable income is more than $100,000, and the filing status is is Married filing separately, we use the Tax Computation Worksheet to figure the tax.

Line 16 - Tax: $146,144 * 0.24 = **$35,074.56**

Figure 8: Failure Case Studies. Existing LLMs commonly fail due to inadequate rule recall, inappropriate usage of similar rules, and computation errors.

LLMs fail to recall certain rules. As discussed in Section 4.2, LLMs often neglect non-essential rules. In airline problems, for instance, a crucial requirement is to apply either the oversize fee or the overweight fee (whichever is higher) and not to sum them. However, LLMs frequently overlook this instruction and incorrectly combine both fees, resulting in an inflated, incorrect total cost.

LLMs get confused by similar rules. When multiple rules appear similar but are applicable under different conditions, LLMs can misapply them. For example, in the NBA domain, teams under the Salary Cap should use the Mid-Level Exception for Room Teams, whereas teams above the Salary Cap should apply the Non-Taxpayer Mid-Level Exception. As illustrated in the second failure case of Figure 8, LLMs sometimes conflate these exceptions. Similar confusion also arises with various Traded Player Exceptions and differing types of Bird Rights.[12]

LLMs compute incorrect results. Mathematical and logical operations present ongoing challenges. For example, in the tax scenario, LLMs must accurately compute a series of values related to income, tax brackets, and credits. Even a minor arithmetic mistake compromises the final result, as shown in the third failure case. Such computational errors underscore the need for more precise and reliable reasoning capabilities in LLMs.

E LLM Prompts

Airline. The prompt template we use in airline domain is as follows.

```
System Prompt: You are a helpful assistant at American Airlines.

User Prompt: You are given the information of a passenger, his / her items, his / her special
needs, and the policies of American Airlines. You should compute the total cost (including
the flight ticket fee, checked bag fees, cost of special needs) according to the policies for
the passenger. The policies of American Airlines are as follows:

<reference_rules>

<user_query> Compute the total cost for him step by step (don't omit any bag) and end your
response with "The total cost is $xxx." (xxx is a number)
Your response:
```

[12]Explanations of these specific NBA terms can be found in the Appendix A.

NBA. The prompt template we use in NBA domain is as follows.

```
System Prompt: You are a helpful NBA team consultant.

User Prompt: You are given rules in NBA Collective Bargaining Agreement and the information
about some teams and players. Then you will be given a list of operations, each of which
desribes how some teams conduct some transaction. You should determine whether each operation
complies with the given rules.

Assume:
* the Salary Cap for the prior (2023-24) Salary Cap Year is $136,000,000;
* the Average Player Salary for the prior (2023-24) Salary Cap Year is $9,700,000;
* the Salary Cap for the current (2024-25) NBA Salary Cap Year is $140,588,000;
* the Luxury Tax is $170,814,000;
* the First Apron Level is $178,132,000;
* the Second Apron Level is $188,931,000;
* the Team Salary of each team listed under "Team Situations:" do not include the amount of
contracts that expire at the end of 2023-2024 Salary Cap Year.

Reference Rules in NBA Collective Bargaining Agreement:

<reference_rules>

Decide whether any operation by any team violate the rules:

<user_query>

Analyze the described operations and explicitly state the type of Salary Cap Exceptions if
you think the exception should be involved. Conclude your response with:
* "Answer: False." if there is no violation to the rules;
* "Answer: True. Illegal Operation: X. Problematic Team: Y." if Team Y in Operation X
violates the rules. Both X and Y should be a single capital letter as A/B/C/...
Your response:
```

Tax. The prompt template we use in tax domain is as follows, where “¡irs_forms¿” includes both form instructions and user query information.

```
System Prompt: You are a helpful US taxation consultant.

User Prompt: You are given several forms used to report US income tax and the instructions or
rules about how to fill the forms. Then you will be given the income and/or payment
information about a tax payer According to the given information. You should calculate the
income tax owed by this payer.

IRS Forms for the tax payer:

<irs_forms>

Calculate the tax owed by the payer step-by-step according to the information provided by the
forms. You should calculate all fields marked with [__]. DO NOT round numbers without
explicit instructions. End your response with:
1. "The total tax owed is $xxx." (xxx is a number) if there is tax owed.
2. "The total tax overpaid is $xxx." (xxx is a number) if there is tax overpaid (and should
be refunded).
Your response:
```